UNIVERSIDAD POLITECNICA SALESIANA

FACULTAD DE INGENIERIAS

SEDE QUITO-CAMPUS SUR

CARRERA DE INGENIERIA DE SISTEMAS

MENCION TELEMATICA

¨ANALISIS Y ESTUDIO DE LAS TECNICAS DE COMPRESION DE DATOS PARA LA INTEGRACION DE LA TECNOLOGIA PDH/SDH SOBRE REDES ETHERNET E IMPLEMENTACION DE UN ALGORITMO DE COMPRESION MEDIANTE SOFTWARE DE SIMULACION¨

TESIS PREVIA A LA OBTENCION DEL TITULO DE INGENIERO DE SISTEMAS

AUTORES

GÁLVEZ CASTILLO JUAN CARLOS

MOROCHO VALLEJO LUIS ROBERTO

DIRECTOR

ING. MARLON CARTAGENA

QUITO, JUNIO del 2010

DECLARACION

Nosotros, Juan Carlos Gálvez Castillo y Luis Roberto Morocho Vallejo, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos los derechos de propiedad intelectual correspondientes a este trabajo, a la Universidad Politécnica Salesiana, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normatividad vigente.

Juan Carlos Gálvez Castillo

Luis Roberto Morocho

CERTIFICACION

Certifico que el presente trabajo fue desarrollado por los Srs. **GALVEZ CASTILLO JUAN CARLOS** y **MOROCHO VALLEJO LUIS ROBERTO** bajo mi dirección.

Ing. Marlon Cartagena

AGRADECIMIENTOS

Al rey de reyes por la sabiduría y el tiempo para culminar un objetivo para mi vida. A mis Padres Milton y Josefina por la vida, el amor, la paciencia en cuidarme, los que inspiraron amor, trabajo, esfuerzo y con su enseñanza de los principios que guían mi camino y su incondicional apoyo a mis estudios. Al Ing. Marlon Cartagena, por el apoyo académico, su amistad y confianza. A Roberto mi compañero y amigo por su ideal de vida y por el gran apoyo profesional en éste documento.

Juan Carlos Gálvez

A Dios por sus bendiciones hacia nosotros y nuestras familias. Nuestro más sincero agradecimiento a la UNIVERSIDAD POLITÉCNICA SALESIANA por habernos brindado su acogida y darnos la oportunidad de culminar nuestra carrera. A nuestro Director Ing. Marlon Cartagena, por brindarnos su valioso tiempo durante la realización de este proyecto. A Juan mi amigo quien a sido un gran aliado durante la vida universitaria

Roberto Morocho

DEDICATORIA

Todo tiene su tiempo….

A Dios el dueño de la vida y el tiempo, quien ha permitido el cumplimiento de esta etapa de mi vida, quien sostiene mi mano para no caer y llena de sabiduría mi mente y corazón. A mi Padre Milton quien es el hombre sabio que orienta mi camino, a mi Madre Josefina con su amor y paciencia impulsó el cumplimiento de cada meta que me propuse. A mis queridos hermanos Fernando, Toño, Katherine y Belén cada uno en su manera de ser me inspira a ser una mejor persona cada día. A mi esposa Verito quien es la dueña de mi amor y corazón. A todos los miembros de mi familia con su granito de arena han formado los pilares que sujetan todos los factores de mi vida.

Juan Carlos Gálvez

A Dios por haberme permitido cruzar los obstáculos y situarme en esta etapa de mi vida; a mis padres Luis y Beatriz por su ejemplo de amor, lucha y superación, a mis hermanos Fernanda y Christian por su apoyo incondicional cuando más lo necesitaba, a Paola mi esposa y Doménica mi hija quienes son el motivo principal y pilar fundamental en mi diario vivir y por quienes lucho incansablemente por llegar lo más lejos posible y tratar de brindarles días provechosos.

Roberto Morocho

ÍNDICE

1. CAPITULO I....19
CONVERGENCIA DE REDES PDH/SDH A ETHERNET....19
1.1 LIMITACIONES DE LAS TECNOLOGÍAS DE LAS REDES CLÁSICAS (TRANSFERENCIA)....19
1.1.1 CLASIFICACIÓN DE REDES POR SU COBERTURA....19
1.1.1.1 Redes de área local (LAN)....20
1.1.1.1.1 Tipos de redes LAN....21
1.1.1.1.1.1 Redes Token Ring....21
1.1.1.1.1.2 Redes Ethernet....23
1.1.1.1.1.3 Redes VLAN....25
1.1.1.1.1.4 Redes WLAN....27
1.1.1.2 Redes de área metropolitana (MAN)....29
1.1.1.3 Redes de área amplia (WAN)....31
1.1.1.3.1 Componentes de una red WAN....31
1.2 MODELO DE REDES....32
1.2.1 INTRODUCCIÓN....32
1.2.2 ESTRUCTURA DEL MODELO OSI DE ISO....33
1.2.2.1 Capas del modelo OSI....34
1.2.3 EL MODELO POR CAPAS DE TCP/IP DE INTERNET....37
1.2.4 DIFERENCIAS Y SIMILITUDES ENTRE OSI y TCP....39
1.2.5 EL MODELO IEEE....40
1.2.5.1 IEEE 802.3....40
1.2.5.2 IEEE 802.5....42
1.3 ARQUITECTURA DE REDES....44
1.3.1 ARQUITECTURA DE RED BASADA EN EL MODELO OSI....45
1.3.2 ARQUITECTURA DE RED X.25....47
1.3.2.1 Características del protocolo X.25....48
1.3.2.2 Capas de funcionalidad X.25....49
1.3.2.3 Ventajas e inconvenientes de X.25....50
1.3.2.4 Limitaciones de la recomendación X.25....50
1.3.3 ARQUITECTURA SNA....50
1.3.4 ARQUITECTURA DE RED DIGITAL (DNA)....52
1.3.5 ARQUITECTURA ARCNET....53
1.3.6 ARQUITECTURA ETHERNET....53
1.3.6.1 Funciones de la arquitectura Ethernet....53
1.4 CONVERGENCIA EN REDES DE DATOS....54

1.4.1 CONVERGENCIA DIGITAL....55
1.4.2 CONVERGENCIA DE REDES Y SERVICIOS....55
1.4.2.1 Convergencia de redes unificadas....56
1.4.2.2 Convergencia a redes IP....56
1.4.3 CONVERGENCIA Y MOVILIDAD....57
1.4.4 CONVERGENCIA PDH/SDH SOBRE ETHERNET....58
1.4.4.1 Antecedentes....58
1.4.4.2 Convergencia de Ethernet sobre PDH....59
1.4.4.3 Convergencia de Ethernet sobre SDH....60
2. CAPITULO II....63
ALGORITMOS DE COMPRESIÓN PARA REDES DE DATOS....63
2.1 GENERALIDADES....63
2.1.1 INTRODUCCIÓN....63
2.1.2 HISTORIA DE LA COMPRESIÓN....63
2.1.3 TIPOS DE COMPRESIÓN....64
2.1.3.1 La compresión física y lógica....64
2.1.3.2 La compresión simétrica y asimétrica....64
2.1.3.3 La compresión reversible e irreversible....65
2.1.3.4 La codificación adaptiva, la semi adaptiva y la no adaptiva....66
2.2 ESTÁNDARES MÁS USADOS EN COMPRESIÓN EN EL DOMINIO IP....67
2.2.1 TIPOS DE DATOS....67
2.2.1.1 Datos multimedia....67
2.2.1.2 Texto....68
2.2.1.3 Imágenes....68
2.2.1.4 Audio....68
2.2.1.5 Video....69
2.2.2 ESTÁNDARES DE COMPRESIÓN DE IMÁGENES....69
2.2.2.1 Estándar de compresión para aplicaciones de fax....70
2.2.2.2 Estándar JBIG....70
2.2.2.3 Estándar JPEG....70
2.2.2.4 Estándar JPEG2000....71
2.2.2.5 Estándar GIF....71
2.2.3 ESTÁNDARES DE COMPRESIÓN DE VIDEO....72
2.2.3.1 Vídeo como una secuencia de imágenes M-JPEG....72
2.2.3.2 MPEG....73
2.2.3.3 H.263....74
2.2.3.4 H.323....74
2.2.4 ESTANDARES DE COMPRESIÓN DE VOZ....75
2.2.4.1 G.711 Modulación por impulsos codificados (MIC) de frecuencias vocales....75

2.2.4.1.1 Introducción 75
2.2.4.2 Estándar G.723.1 códec de voz de doble velocidad para la transmisión en comunicaciones multimedios a 5,3 Y 6,3 kbit/s 80
2.2.4.2.1 Introducción 80
2.2.4.3 Estándar G.728 codificación de señales vocales a 16 kbit/s 81
2.2.4.3.1 Introducción 81
2.3 ESTRUCTURA 82
2.3.2 ALGORITMO DE HUFFMAN 83
2.4 VENTAJA Y DESVENTAJA DE LA COMPRESIÓN 86
3. CAPITULO III 88
TECNOLOGÍA PDH 88
3.1 TÉCNICAS DE TRANSMISIÓN 88
3.1.1 TRANSMISIÓN SÍNCRONA 88
3.1.2 TRANSMISIÓN ASÍNCRONA 89
3.2 ESTÁNDARES DE MULTIPLEXACIÓN PLESIÓCRONA Y SÍNCRONA 90
3.2.1 MULTIPLEXACIÓN POR DIVISIÓN DE ESPACIO (SDM) 91
3.2.2 MULTIPLEXACIÓN POR DIVISIÓN DE FRECUENCIA (FDM) 92
3.2.3 MULTIPLEXACIÓN POR DIVISIÓN DE TIEMPO (TDM) 94
3.2.3.1 Bit-Interleaved Multiplexing 95
3.2.3.2 Byte-Interleaved Multiplexing 96
3.3 JERARQUÍAS DE MULTIPLEXACIÓN 96
3.3.1 JERARQUÍA EUROPEA 97
3.3.2 JERARQUÍA NORTEAMERICANA 98
3.3.3 JERARQUÍA JAPONESA 98
3.4 NORMA EUROPEA ETSI 98
3.4.1 PRIMER ORDEN JERÁRQUICO 98
3.4.2 SEGUNDO ORDEN JERÁRQUICO 99
3.4.3 TERCER ORDEN JERÁRQUICO 99
3.4.4 CUARTO ORDEN JERÁRQUICO 100
3.5 GESTIÓN DE REDES PDH 100
3.5.1. FUNCIONES DE LA TELESUPERVISIÓN PDH 101
3.5.1.1 Estación remota 102
3.5.1.2 Estación central 104
3.5.1.2 Software de telesupervisión 104
3.6 VENTAJAS Y DESVENTAJAS 105
3.6.1 VENTAJAS 105
3.6.2 DESVENTAJAS 106
4. CAPITULO IV 107
TECNOLOGÍA SDH 107

4.1 ESTRUCTURA DE LA TRAMA SDH 107
4.1.1 INTRODUCCIÓN 107
4.1.2 CARACTERÍSTICAS SDH 107
4.1.3 ESTRUCTURA BASICA DE SDH 108
4.1.3.1 Terminología SDH 109
4.2 ESTRUCTURA DE LA TRAMA STM-1, STM-n 110
4.2.1 TRAMA SDH STM-1 110
4.2.1.1 Sección de cabecera SOH 111
4.2.1.2 Cabecera de direcciones 111
4.2.1.3 Punteros 112
4.2.2 TRAMA SDH STM-N 112
4.3 ARQUITECTURA FUNCIONAL DE LA RED SDH 113
4.3.1 EQUIPAMIENTO SDH 113
4.3.1.1 Regenerador 113
4.3.1.2 Multiplexor terminal 114
4.3.1.3 Multiplexores sumadores 114
4.3.1.4 Conmutadores de cruce de conexión 115
4.3.2 ARQUITECTURA DE RED SDH 115
4.3.2.1 Punto a Punto 115
4.3.2.2 Lineales 116
4.3.2.3 Anillos 116
4.4 MULTIPLEXACIÓN SDH 117
4.4.1 JERARQUIAS DE MULTIPLEXACIÓN SDH 118
4.5 SINCRONISMO EN REDES SDH 121
4.5.1 METODOS DE SINCRONIZACIÓN 121
4.5.2 SINCRONISMO INTRA ESTACIÓN 123
4.5.2.1 Sincronismo de equipos jerarquías PDH 123
4.5.2.2 Sincronismo de equipos jerarquías SDH 123
4.5.2.3 Programación del sincronismo 123
4.5.3 SINCRONISMO INTER-ESTACIÓN 124
4.5.3.1 Propagación del sincronismo en PDH 124
4.5.3.2 Propagación del sincronismo en SDH 125
4.5.3.3 Loop de sincronismo 125
4.6 GESTIÓN DE REDES SDH 126
4.6.1 COMPONENTES DE LA GESTION SDH 127
4.6.1.1 Unidad de Control y Gestión 128
4.6.1.2 Canal de comunicación hacia la PC en la estación local 129
4.6.1.3 Red de comunicación entre distintos equipos en una misma estación 129
4.6.1.4 Red de comunicación en el Centro de Gestión Regional 129

4.6.1.5 Red de comunicación entre Centros Regionales con el Centro Nacional Unificado 130
4.6.1.6 Software de Aplicación 130
4.6 VENTAJAS Y DESVENTAJAS 130
5. CAPITULO V 132
DESARROLLO DEL SOFTWARE DE SIMULACIÓN....132
5.1 ELECCIÓN DE LA HERRAMIENTA....132
5.2 MODELO DEL ALGORITMO DE COMPRESIÓN 132
5.3 DISEÑO DEL ALGORITMO DE COMPRESION 135
5.3.1 DIAGRAMAS DE CASOS DE USO....135
5.3.2 DIAGRAMA DE MODELO DE DOMINIO 144
5.3.3 DICCIONARIO DE DATOS 147
5.4 FASE DE PRUEBAS Y CORRECCIONES 166
5.5 SIMULACIÓN EN LA RED....167
5.6 MANUAL DE USUARIO 167
6. CAPITULO VI 212
CONCLUSIONES Y RECOMENDACIONES 212
6.1 CONCLUSIONES 212
6.2 RECOMENDACIONES 215
REFERENCIAS BIBLIOGRAFICAS....217
GLOSARIO DE TERMINOS....218
ANEXOS 225
Anexo A. Código Fuente....227
Anexo B. Cálculos estadísticos del algoritmo de compresión. 275

INDICE DE FIGURAS

FIG. 1. 1. ESQUEMA DE UNA RED LAN 21
FIG. 1. 2. MODELO DE UNA RED TOKEN RING 22
FIG. 1. 3. DIAGRAMA DE INTERCONEXIÓN BUS 23
FIG. 1. 4. DIAGRAMA DE ETHERNET ESTRELLA 24
FIG. 1. 5. DIAGRAMA DE ETHERNET ANILLO 24
FIG. 1. 6. DIAGRAMA DE ETHERNET ÁRBOL 25
FIG. 1. 7. DIAGRAMA DE ETHERNET MALLA 25
FIG. 1. 8. MODELO DE UNA RED VLAN 26
FIG. 1. 9. MODELO DE UNA RED AD-HOC 28
FIG. 1. 10. MODELO DE UNA RED INFRAESTRUCTURE 28
FIG. 1. 11. ESQUEMA DE UNA RED WLAN 29
FIG. 1. 12. ESQUEMA DE UNA RED MAN 30
FIG. 1. 13. ESQUEMA DE UNA RED WAN 32
FIG. 1. 14. ESQUEMA DE LA TRAMA HDLC 35
FIG. 1. 15. ARQUITECTURA DE LA RED BASADA EN EL MODELO OSI 37
FIG. 1. 16. MODELO DE CAPAS TCP 38
FIG. 1. 17. FORMATO DE TRAMA 802.3 41
FIG. 1. 18. FORMATO DE TRAMA 802.5 43
FIG. 1. 19. MODELO DE REFERENCIA OSI 46
FIG. 1. 20. ARQUITECTURA DE RED X.25 48
FIG. 1. 21. ARQUITECTURA DE RED SNA 51
FIG. 1. 22. ARQUITECTURA DE RED DNA 52
FIG. 1. 23. MODELO DE CONVERGENCIA 55
FIG. 1. 24. IMPORTANCIA DEL PROTOCOLO IP EN LA CONVERGENCIA 57
FIG. 1. 25. ESTRUCTURA DE TRAMA GFP 61

FIG. 2. 1. COMPRESIÓN DE DATOS 67
FIG. 2. 2. COMPARACIÓN JPEG – MPEG-4 74
FIG. 2. 3. MUESTREO DE UNA SEÑAL ANALÓGICA 76
FIG. 2. 4. CUANTIFICACIÓN DE UNA SEÑAL 76
FIG. 2. 5. LEY A 78
FIG. 2. 6. LEY DE CODIFICACIÓN LEY A Y LEY M 79
FIG. 2. 7. PALABRA MIC CODIFICADA. 79

FIG. 3. 1. MODO DE TRANSMISIÓN ASÍNCRONA 89
FIG. 3. 2. COMPONENTES DE UN MULTIPLEXOR 91
FIG. 3. 3. MULTIPLEXACIÓN POR DIVISIÓN DE FRECUENCIA 92
FIG. 3. 4. INTERFERENCIA CO-CANAL 93
FIG. 3. 5. INTERFERENCIA POR CANAL ADYACENTE 93
FIG. 3. 6. MULTIPLEXACIÓN POR DIVISIÓN DE TIEMPO 95
FIG. 3. 7. DIGITALIZACIÓN DE UN CANAL DE VOZ ANALÓGICO 97
FIG. 3. 8. JERARQUÍA EUROPEA DE MULTIPLEXACIÓN 97
FIG. 3. 9. JERARQUÍA NORTEAMERICANA DE MULTIPLEXACIÓN 98
FIG. 3. 10. JERARQUÍA JAPONESA DE MULTIPLEXACIÓN 98
FIG. 3. 11. PRIMER ORDEN JERÁRQUICO 99
FIG. 3. 12. SEGUNDO ORDEN JERÁRQUICO 99
FIG. 3. 13. TERCER ORDEN JERÁRQUICO 100
FIG. 3. 14. CUARTO ORDEN JERÁRQUICO 100
FIG. 3. 15. ESTRUCTURA DE UNA RED DE TELESUPERVISIÓN 102

FIG. 4. 1. ESTRUCTURA TRAMA STM-1 108
FIG. 4. 2. MAPA DE ESTRUCTURA SDH 110

FIG. 4. 3. POSIBILIDAD DE CARGA VC4 111
FIG. 4. 4. TRAMA STM-N 112
FIG. 4. 5. ELEMENTOS DE UNA RED SDH 113
FIG. 4. 6. BLOQUE DEL REGENERADOR 114
FIG. 4. 7. BLOQUE DEL MULTIPLEXOR TERMINAL 114
FIG. 4. 8. BLOQUE DEL MULTIPLEXOR ADD/DROP 114
FIG. 4. 9. BLOQUE DEL MULTIPLEXOR CROSSCONECTOR 115
FIG. 4. 10. TOPOLOGÍA PUNTO A PUNTO 116
FIG. 4. 11. APLICACIÓN LINEAL 116
FIG. 4. 12. APLICACIÓN EN ANILLO 116
FIG. 4. 13. MULTIPLEXACIÓN DE PRIMER ORDEN 118
FIG. 4. 14. MULTIPLEXACIÓN DE SEGUNDO ORDEN 119
FIG. 4. 15. MULTIPLEXACIÓN DE TERCER ORDEN 119
FIG. 4. 16. MULTIPLEXACIÓN DE CUARTO ORDEN 120
FIG. 4. 17. MULTIPLEXACIÓN DE QUINTO ORDEN 120
FIG. 4. 18. MAPA DE SINCRONIZACIÓN 122
FIG. 4. 19. ESTRUCTURA DE ENLACE PARA GESTIÓN DE EQUIPOS SDH 128

FIG. 5. 1 DIAGRAMA DE CASOS DE USO 136
FIG. 5. 2 DIAGRAMA DE SECUENCIA ELEGIR TECNOLOGÍA DE TX 137
FIG. 5. 3 DIAGRAMA DE CLASES ELEGIR TECNOLOGÍA DE TX 138
FIG. 5. 4 DIAGRAMA DE SECUENCIA AGREGAR TRAMAS 139
FIG. 5. 5 DIAGRAMA DE CLASES AGREGAR TRAMAS 139
FIG. 5. 6 DIAGRAMA DE SECUENCIA AGREGAR USUARIOS 140
FIG. 5. 7 DIAGRAMA DE CLASES AGREGAR USUARIOS 141
FIG. 5. 8 DIAGRAMA DE SECUENCIA AGREGAR CANALES 142
FIG. 5. 9 DIAGRAMA DE CLASES AGREGAR CANALES 142
FIG. 5. 10 DIAGRAMA DE SECUENCIA CODIFICAR 143
FIG. 5. 11 DIAGRAMA DE CLASES CODIFICAR HUFFMAN 144
FIG. 5. 12 DIAGRAMA DE MODELO DE DOMINIO 145
FIG. 5. 13 DIAGRAMA DE CLASES 146
FIG. 5. 14. FORMULARIO PRINCIPAL 147
FIG. 5. 15. ASIGNACIÓN DE LA INFORMACIÓN EN UNA TRAMA E1 148
FIG. 5. 16 INTERFAZ DE VARIOS USUARIOS INTERACTUANDO CON UNA TRAMA E1 149
FIG. 5. 17 INTERFAZ DE VARIOS USUARIOS INTERACTUANDO CON VARIOS E1 150
FIG. 5. 18 INTERFAZ DE VARIOS USUARIOS INTERACTUANDO CON UNA TRAMA STM-1 A NIVEL DE TU-12 151
FIG. 5. 19 CANAL DE UNA TRAMA 152
FIG. 5. 20 CONTENEDOR DE TRAMAS STM-1 A NIVEL DE TU-12 153
FIG. 5. 21 INTERFAZ DEL CONTENEDOR DE USUARIOS QUE INTERACTÚAN CON TRAMAS STM-1 A NIVEL DE TU-12 154
FIG. 5. 22 INTERFAZ DE USUARIOS QUE INTERACTÚAN CON TRAMAS E1, STM-1 A NIVEL VC-3 Y VC-4 155
FIG. 5. 23 CONTROL QUE CONTIENE LA ESTRUCTURA DE LA TRAMA E1 156
FIG. 5. 24 INTERFAZ DEL CONTENEDOR DE TRAMAS 156
FIG. 5. 25 INTERFAZ DEL CONTENEDOR DE TRAMAS E1 157
FIG. 5. 26 INTERFAZ DE LA AGRUPACIÓN DE CANALES TU12 158
FIG. 5. 27 INTERFAZ DE USUARIO 159
FIG. 5. 28 INTERFAZ DE USUARIO PARA SELECCIÓN DE ANCHO DE BANDA 160
FIG. 5. 29 REPORTE GRÁFICO DE LAS ESTADÍSTICAS DE COMPRESIÓN EN FUNCIÓN DEL TIEMPO 164
FIG. 5. 30 RESUMEN ESTADÍSTICO DE LOS CÁLCULOS DE LOS TIEMPOS DE TRANSMISIÓN 164
FIG. 5. 31 REPORTE GRÁFICO DE LAS ESTADÍSTICAS DE COMPRESIÓN EN FUNCIÓN DEL TAMAÑO DE ARCHIVO 165
FIG. 5. 32 VISTA DE LA BASE DE DATOS DONDE SE ALMACENAN LAS VARIABLES QUE INTERVIENEN EN LOS CÁLCULOS DE COMPRESIÓN. 165
FIG. 5. 33 ICONO DE INSTALACIÓN 168
FIG. 5. 34 ADVERTENCIA DE SEGURIDAD 169
FIG. 5. 35 PROCESO DE INSTALACIÓN 169
FIG. 5. 36 PANTALLA INICIAL 169

FIG. 5. 37 VERIFICANDO LA INSTALACIÓN DE LA APLICACIÓN 170
FIG. 5. 38 PROCESO DE DESINSTALACIÓN DE LA APLICACIÓN 170
FIG. 5. 39 QUITANDO EL PROGRAMA HUFFMAN 171
FIG. 5. 40 PROCESO DE DESINSTALACIÓN DE LA APLICACIÓN 171
FIG. 5. 41 OPCIÓN SALIR DE LA APLICACIÓN 172
FIG. 5. 42 OPCIÓN ARCHIVO EN E1 172
FIG. 5. 43 INTERFAZ DE ASIGNACIÓN DE INFORMACIÓN A CANALES DE UNA TRAMA E1 173
FIG. 5. 44 SELECCIÓN DEL ARCHIVO DE PRUEBA 173
FIG. 5. 45 ASIGNACIÓN DE LA INFORMACIÓN EN LOS CANALES DE LA TRAMA E1 174
FIG. 5. 46 OPCIÓN VARIOS USUARIOS INTERACTUANDO CON UN E1 174
FIG. 5. 47 INTERFAZ DE USUARIO PARA LA SELECCIÓN DE LOS PARÁMETROS DE TRANSMISIÓN 175
FIG. 5. 48 INGRESO DE LAS VARIABLES QUE INTERVIENEN EN LA TRANSMISIÓN 175
FIG. 5. 49 ANIMACIÓN QUE EJEMPLIFICA LA FORMACIÓN DE UNA TRAMA E1 175
FIG. 5. 50 FORMACIÓN DE UNA TRAMA E1 176
FIG. 5. 51 SELECCIÓN DE LOS CANALES DEL ANCHO DE BANDA 176
FIG. 5. 52 VISTA GENERAL 176
FIG. 5. 53 ELECCIÓN DE LOS ARCHIVOS QUE INTERVIENEN EN EL PROCESO DE LA COMPRESIÓN 177
FIG. 5. 54 VISTA DE LA INTERFAZ DE USUARIO, ELECCIÓN DE ARCHIVOS A COMPRIMIR 178
FIG. 5. 55 ESTADÍSTICAS DE COMPRESIÓN 178
FIG. 5. 56 SE EXPORTA A EXCEL LAS ESTADÍSTICAS OBTENIDAS LUEGO DE LA COMPRESIÓN 179
FIG. 5. 57 SE EXPORTA A PDF LAS ESTADÍSTICAS OBTENIDAS LUEGO DE LA COMPRESIÓN 179
FIG. 5. 58 RESULTADOS EXPORTADOS A FORMATO EXCEL 179
FIG. 5. 59 RESULTADOS EXPORTADOS A PDF 180
FIG. 5. 60 REFRESCAR LOS DATOS DE LA TABLA 180
FIG. 5. 61 AJUSTAR EL TAMAÑO DE VISUALIZACIÓN 180
FIG. 5. 62 OPCIÓN VARIOS USUARIOS INTERACTUANDO CON VARIOS E1 181
FIG. 5. 63 INTERFAZ DE USUARIO PARA LA SELECCIÓN DE LOS PARÁMETROS DE TRANSMISIÓN 181
FIG. 5. 64 INGRESO DE LAS VARIABLES QUE INTERVIENEN EN LA TRANSMISIÓN 182
FIG. 5. 65 FORMACIÓN DE UNA TRAMA E1 DINÁMICA 182
FIG. 5. 66 SELECCIÓN DE LOS CANALES DEL ANCHO DE BANDA 182
FIG. 5. 67 SELECCIÓN DE LAS VARIABLES DE TRANSMISIÓN 183
FIG. 5. 68 AGREGAR USUARIOS 183
FIG. 5. 69 SELECCIÓN DE LOS CANALES DEL ANCHO DE BANDA 184
FIG. 5. 70 ELECCIÓN DE LOS ARCHIVOS QUE INTERVIENEN EN EL PROCESO DE LA COMPRESIÓN 185
FIG. 5. 71 VISTA DE LA INTERFAZ DE USUARIO, ELECCIÓN DE ARCHIVOS A COMPRIMIR 185
FIG. 5. 72 ESTADÍSTICAS DE COMPRESIÓN 186
FIG. 5. 73 SE EXPORTA A EXCEL LAS ESTADÍSTICAS OBTENIDAS LUEGO DE LA COMPRESIÓN 186
FIG. 5. 74 SE EXPORTA A PDF LAS ESTADÍSTICAS OBTENIDAS LUEGO DE LA COMPRESIÓN 187
FIG. 5. 75 RESULTADOS EXPORTADOS A PDF 187
FIG. 5. 76 RESULTADOS EXPORTADOS A FORMATO EXCEL 187
FIG. 5. 77 REFRESCAR LOS DATOS DE LA TABLA 188
FIG. 5. 78 AJUSTAR EL TAMAÑO DE VISUALIZACIÓN 188
FIG. 5. 79 OPCIÓN VARIOS USUARIOS INTERACTUANDO CON VARIOS E1 189
FIG. 5. 80 INTERFAZ DE USUARIO PARA LA SELECCIÓN DE LOS PARÁMETROS DE TRANSMISIÓN 189
FIG. 5. 81 INGRESO DE LAS VARIABLES QUE INTERVIENEN EN LA TRANSMISIÓN 189
FIG. 5. 822 FORMACIÓN DE UNA TRAMA STM-1 A NIVEL DE TU-12 190
FIG. 5. 83 INTERFAZ DE USUARIO INTERACTUANDO CON UNA TRAMA STM-1 A NIVEL DE CANALES TU-12 190
FIG. 5. 84 ANIMACIÓN QUE EJEMPLIFICA LA FORMACIÓN DE UNA TRAMA STM-1 PARTIENDO DE UN E1 191
FIG. 5. 85 INGRESO DE LAS VARIABLES QUE INTERVIENEN EN LA TRANSMISIÓN 191
FIG. 5. 86 SE AGREGAN LOS CANALES DE UNA TRAMA STM-1 192
FIG. 5. 87 ELECCIÓN DEL CANAL Y LOS ANCHOS DE BANDA 192
FIG. 5. 88 CANALES DE LA TRAMA STM-1 QUE HAN SIDO SELECCIONADOS 193
FIG. 5. 89 ELECCIÓN DE LOS ARCHIVOS QUE INTERVIENEN EN EL PROCESO DE LA COMPRESIÓN 193
FIG. 5. 90 VISTA DE LA INTERFAZ DE USUARIO, ELECCIÓN DE ARCHIVOS A COMPRIMIR 194
FIG. 5. 91 VISUALIZACIÓN DE LAS ESTADÍSTICAS DE COMPRESIÓN 194
FIG. 5. 92 SE EXPORTA A EXCEL LAS ESTADÍSTICAS OBTENIDAS LUEGO DE LA COMPRESIÓN 195

FIG. 5. 93 SE EXPORTA A PDF LAS ESTADÍSTICAS OBTENIDAS LUEGO DE LA COMPRESIÓN 195
FIG. 5. 94 RESULTADOS EXPORTADOS A PDF 195
FIG. 5. 95 RESULTADOS EXPORTADOS A FORMATO EXCEL 196
FIG. 5. 96 REFRESCAR LOS DATOS DE LA TABLA 196
FIG. 5. 97 AJUSTAR EL TAMAÑO DE VISUALIZACIÓN 196
FIG. 5. 98 OPCIÓN VARIOS USUARIOS STM-1 INTERACTUANDO CON CANALES VC-3 197
FIG. 5. 99 INTERFAZ DE USUARIO PARA LA SELECCIÓN DE LOS PARÁMETROS DE TRANSMISIÓN 197
FIG. 5. 100 INGRESO DE LAS VARIABLES QUE INTERVIENEN EN LA TRANSMISIÓN 198
FIG. 5. 101 FORMACIÓN DE UNA TRAMA STM-1 DINÁMICA 198
FIG. 5. 102 SELECCIÓN DE LOS CANALES QUE INTERVIENEN EN LA SIMULACIÓN 198
FIG. 5. 103 SELECCIÓN DE LAS VARIABLES DE TRANSMISIÓN 199
FIG. 5. 104 AGREGAR USUARIOS 199
FIG. 5. 105 SELECCIÓN DE LOS CANALES DEL ANCHO DE BANDA 200
FIG. 5. 106 ELECCIÓN DE LOS ARCHIVOS QUE INTERVIENEN EN EL PROCESO DE LA COMPRESIÓN 201
FIG. 5. 107 VISTA DE LA INTERFAZ DE USUARIO, ELECCIÓN DE ARCHIVOS A COMPRIMIR 201
FIG. 5. 108 ESTADÍSTICAS DE COMPRESIÓN 202
FIG. 5. 109 SE EXPORTA A EXCEL LAS ESTADÍSTICAS OBTENIDAS LUEGO DE LA COMPRESIÓN 202
FIG. 5. 110 SE EXPORTA A PDF LAS ESTADÍSTICAS OBTENIDAS LUEGO DE LA COMPRESIÓN 202
FIG. 5. 111 SE EXPORTA A PDF LAS ESTADÍSTICAS OBTENIDAS LUEGO DE LA COMPRESIÓN 203
FIG. 5. 112 RESULTADOS EXPORTADOS A FORMATO EXCEL 203
FIG. 5. 113 REFRESCAR LOS DATOS DE LA TABLA 203
FIG. 5. 114 AJUSTAR EL TAMAÑO DE VISUALIZACIÓN 204
FIG. 5. 115 OPCIÓN VARIOS USUARIOS INTERACTUANDO CON VARIOS VC-4 204
FIG. 5. 116 INTERFAZ DE USUARIO PARA LA SELECCIÓN DE LOS PARÁMETROS DE TRANSMISIÓN 205
FIG. 5. 117 INGRESO DE LAS VARIABLES QUE INTERVIENEN EN LA TRANSMISIÓN 205
FIG. 5. 118 FORMACIÓN DE UNA TRAMA STM-1 A NIVEL DE TU-12 205
FIG. 5. 119 INTERFAZ DE USUARIO INTERACTUANDO CON UNA TRAMA STM-1 A NIVEL DE CANALES VC-4 206
FIG. 5. 120 INGRESO DE LAS VARIABLES QUE INTERVIENEN EN LA TRANSMISIÓN 206
FIG. 5. 121 SE AGREGAN LOS CANALES DE UNA TRAMA STM-1 207
FIG. 5. 122 ELECCIÓN DEL CANAL Y LOS ANCHOS DE BANDA 208
FIG. 5. 123 ELECCIÓN DE LOS ARCHIVOS QUE INTERVIENEN EN EL PROCESO DE LA COMPRESIÓN 208
FIG. 5. 124 VISTA DE LA INTERFAZ DE USUARIO, ELECCIÓN DE ARCHIVOS A COMPRIMIR 209
FIG. 5. 125 VISUALIZACIÓN DE LAS ESTADÍSTICAS DE COMPRESIÓN 209
FIG. 5. 126 SE EXPORTA A EXCEL LAS ESTADÍSTICAS OBTENIDAS LUEGO DE LA COMPRESIÓN 210
FIG. 5. 127 SE EXPORTA A PDF LAS ESTADÍSTICAS OBTENIDAS LUEGO DE LA COMPRESIÓN 210
FIG. 5. 128 RESULTADOS EXPORTADOS A PDF 210
FIG. 5. 129 RESULTADOS EXPORTADOS A FORMATO EXCEL 211
FIG. 5. 130 REFRESCAR LOS DATOS DE LA TABLA 211
FIG. 5. 131 AJUSTAR EL TAMAÑO DE VISUALIZACIÓN 211

INDICE DE TABLAS

TABLA 2.1 EJEMPLO DE CODIFICACIÓN POR FRECUENCIA 65
TABLA 2.2. GENERACIÓN DE CÓDIGOS EN BASE AL ALGORITMO DE CODIFICACIÓN DE HUFFMAN 83
TABLA 2.3. TABLA DE CÓDIGOS EN BASE AL ALGORITMO DE CODIFICACIÓN DE HUFFMAN 85
TABLA 3.1. JERARQUÍAS DE MULTIPLEXACIÓN EN PDH 97
TABLA 4.1 COMPARACIÓN DE TASAS DE SDH Y SONET 109
TABLA 4.2 COMPARACIÓN ENTRE SISTEMAS DE GESTIÓN PDH Y SDH 126
TABLA 5.1 TABLA DE CÓDIGOS EN BASE AL ALGORITMO DE CODIFICACIÓN HUFFMAN 133
TABLA 5.2 GENERACIÓN DE CÓDIGOS EN BASE AL ALGORITMO HUFFMAN 134
TABLA 5.3 MENÚS DEL FORMULARIO PRINCIPAL 147
TABLA 5.4 VARIABLES DE LA CLASE FORM1 148
TABLA 5.5 FUNCIONES DE LA CLASE FORM1 148
TABLA 5.6 VARIABLES DE LA CLASE FORM2 149
TABLA 5.7 FUNCIONES DE LA CLASE FORM2 149
TABLA 5.8 VARIABLES DE LA CLASE FRMVARIOSE1V2 150
TABLA 5.9 FUNCIONES DE LA CLASE FRMVARIOSE1V2 150
TABLA 5.10 VARIABLES DE LA CLASE STM_TU12 151
TABLA 5.11 FUNCIONES DE LA CLASE STM_TU12 151
TABLA 5.12 VARIABLES CLASE CANAL 152
TABLA 5.13 FUNCIONES DE LA CLASE CANAL 152
TABLA 5.14 VARIABLES DE LA CLASE CTNTU12 153
TABLA 5.15 FUNCIONES DE LA CLASE CNTTU12 153
TABLA 5.16 VARIABLES DE LA CLASE CTNUSERTU12 154
TABLA 5.17 FUNCIONES DE LA CLASE CTNUSERTU12 154
TABLA 5.18 VARIABLES DE LA CLASE CTNUSUARIOS 155
TABLA 5.19 FUNCIONES DE LA CLASE CTNUSUARIOS 155
TABLA 5.20 VARIABLES DE LA CLASE E1 156
TABLA 5.21 FUNCIONES DE LA CLASE E1 156
TABLA 5.22 VARIABLES DE LA CLASE MEDIOS 157
TABLA 5.23 FUNCIONES DE LA CLASE MEDIOS 157
TABLA 5.24 VARIABLES DE LA CLASE MEDIOSV2 157
TABLA 5.25 FUNCIONES DE LA CLASE MEDIOSV2 158
TABLA 5.26 VARIABLES DE LA CLASE TU-12 158
TABLA 5.27 FUNCIONES DE LA CLASE TU-12 158
TABLA 5.28 VARIABLES DE LA CLASE USERE1 159
TABLA 5.29 FUNCIONES DE LA CLASE USERE1 159
TABLA 5.30 VARIABLES DE LA CLASE USUARIO001 160
TABLA 5.31 FUNCIONES DE LA CLASE USUARIO001 160
TABLA 5.32 VARIABLES DE LA CLASE HUFFMAN 161
TABLA 5.33 FUNCIONES DE LA CLASE HUFFMAN 161
TABLA 5.34 VARIABLES DE LA CLASE LIMITARCANALES 162
TABLA 5.35 FUNCIONES DE LA CLASE LIMITARCANALES 162
TABLA 5.36 FUNCIONES DE LA CLASE PROGRAM 162
TABLA 5.37 VARIABLES DE LA CLASE RAMANODO 163
TABLA 5.38 FUNCIONES DE LA CLASE RAMANODO 163
TABLA 5.39 CAMPOS QUE CONFORMAN LA BASE DE DATOS 166

INDICE DE ANEXOS

Anexo A – CODIGO FUENTE

Anexo B – CALCULOS ESTADISTICOS DEL ALGORITMO DE COMPRESIÓN

RESUMEN

El presente Proyecto de Grado, titulado: Análisis y estudio de las técnicas de compresión de datos para la integración de la tecnología PDH/SDH sobre redes Ethernet e implementación de un algoritmo de compresión mediante software de simulación, comprende un estudio investigativo de las características, ventajas y desventajas de las técnicas de compresión de datos, comprende también de la implementación de un algoritmo de compresión que demuestre mediante un software de simulación dichas ventajas o desventajas.

Las comunicaciones son una herramienta importante en el diario convivir de los seres vivos y esto se evidencia claramente desde tiempos remotos hasta nuestros días. Desde el inicio se ha evidenciado grandes desarrollos en las telecomunicaciones y cada vez existe una mayor exigencia para mejorar las condiciones de esta, es por eso que día a día se están desarrollando nuevas tecnologías que ayuden a satisfacer las necesidades de comunicación del ser humano de una manera eficiente e integradora.

En estos momentos, donde la tecnología avanza a pasos acelerados es imprescindible disponer de redes que puedan soportar altas velocidades y de esta manera brindar servicios de calidad para la satisfacción del cliente. Por consiguiente, en este proyecto se realizará un estudio de las redes de transporte PDH y SDH, estas últimas se encuentran constituidas principalmente por fibra óptica, es por ello su alta velocidad y gran confiabilidad, seguidamente se estudiará la tecnología Ethernet y la tendencia que se está viviendo actualmente por llegar a obtener una convergencia de redes y servicios; se analizará las diferentes técnicas de compresión de datos para finalmente aplicar un algoritmo de compresión mediante una aplicación de software de simulación que complemente este tema de estudio.

Se espera alcanzar a cubrir todas las expectativas que se puedan presentar acerca de las redes de transporte PDH, SDH y la interacción con la tecnología Ethernet, se analizará los elementos y topologías que se utilizarán en la

implementación de la misma, para posteriormente realizar el estudio más detallado de de la formación de las tramas básicas PDH y SDH, E1 y STM-1 respectivamente y cada una de las etapas de multiplexación, tanto en PDH como en SDH.

1. CAPITULO I

CONVERGENCIA DE REDES PDH/SDH A ETHERNET

1.1 LIMITACIONES DE LAS TECNOLOGÍAS DE LAS REDES CLÁSICAS (TRANSFERENCIA)

Una red de datos es un sistema compuesto por un conjunto de computadoras, equipos de comunicaciones y otros dispositivos que se comunican entre sí; de esta manera, se comparte información y recursos.

Básicamente una red de datos está integrada por diversos elementos:

- **Los servidores:** En los cuales se encuentra y procesa la información.
- **Los equipos de telecomunicaciones:** (Hubs, switch, módems, routers, etc.).
- **El Patch Panel:** Los cuales son unos organizadores de cables y se interconectan mediante Patch cords.
- **El cableado horizontal y vertical:** Por lo general cable UTP o fibra óptica.
- **Los equipos terminales:** Computadores, impresoras, etc.

1.1.1 CLASIFICACIÓN DE REDES POR SU COBERTURA

Se debe considerar que una verdadera clasificación de las redes de información debe tener en cuenta aspectos tecnológicos como las prestaciones que ofrecen, el tratamiento que dan a la información o su funcionalidad antes que la cobertura que puedan ofrecer, toda vez que los parámetros de distancia permiten verificar o certificar que se cumple con ciertas normas o estándares antes que determinar si una red puede o no tener operatividad; sin embargo, se presenta un ensayo de clasificación de las redes desde el punto de vista de la distancia, para situar el presente estudio utilizando la terminología con la que más se familiariza el lector:

- Redes LAN
- Redes MAN
- Redes WAN

1.1.1.1 Redes de área local (LAN)

Una red de área local (LAN), es la interconexión de varios computadores y periféricos para compartir recursos e intercambiar datos y aplicaciones. Su extensión se limita físicamente a un edificio o a un entorno de alrededor de 1 Km.

Entre las principales características de las redes LAN se pueden resumir en las siguientes:

- Entornos de pocos Kilómetros: Normalmente las LAN ocupan redes de entorno no más de 1 Km.
- Uso de un medio de comunicación privado.
- Altas velocidades de transmisión: Las velocidades de las LAN han ido evolucionando desde los 16 MHz con UTP cat3, hasta 1Gbps con cat6 y 10 Gbps con fibra óptica en la actualidad.
- Gran variedad y número de dispositivos conectados.

Ventajas de las redes locales:

- Compartir programas y archivos.
- Compartir recursos de la red.
- Posibilidad de utilizar software de red.
- Gestión centralizada.
- Acceso a otros sistemas operativos.
- Facilidad de utilización para el usuario.

Los elementos que componen una red son:

- **Servidor:** Computador principal que proporciona servicios a los terminales conectados a la red.
- **Estaciones de trabajo:** Son los equipos terminales de la red.
- **Tarjetas de Red:** Elemento interfaz que permite conexión de un computador a la red.
- **El medio de transmisión:** Conformado por el cableado y los conectores que enlazan los componentes de la red.
- **Dispositivos adicionales:** Son todos aquellos componentes de la red que son utilizados y compartidos por los usuarios de la red; son recursos tales como: impresoras, unidades de almacenamiento, etc.

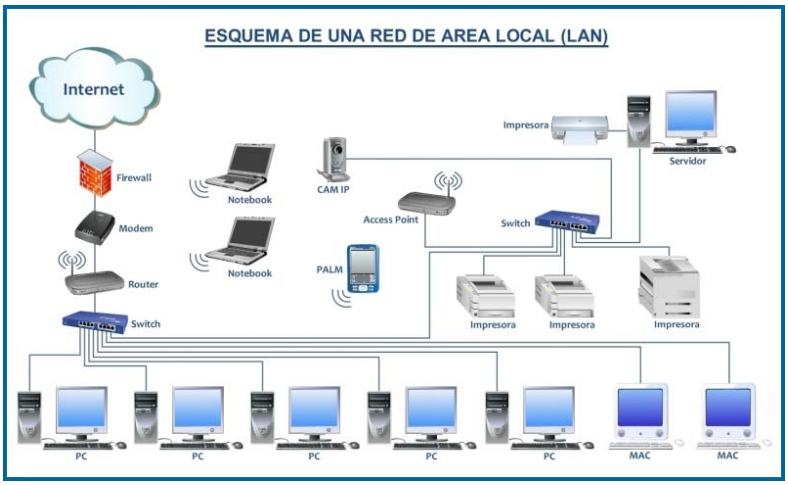


Fig. 1. 1. Esquema de una red LAN [1]

1.1.1.1.1 Tipos de redes LAN

1.1.1.1.1.1 Redes Token Ring

La red Token Ring es una implementación del estándar IEEE 802.5, el cual se distingue más por su método de transmitir la información que por la forma en que se conectan las computadoras.

[1] http://www.sertecom.cl/redes.htm

Su funcionamiento consiste en un Token que es pasado de computador a computador.

Cuando un computador desea enviar información debe de esperar que le llegue el Token vacío, luego utiliza el Token para enviar la información a otro computador, cuando el otro computadora recibe la información regresa el Token al computador que envió con el mensaje de recibido.

Al final se libera el Token para volver a ser usado por cualquier otro computador. Aquí no hay colisiones, el problema implica en el tiempo que debe esperar una computadora para obtener el Token sin utilizar.

Para que una red Token Ring funcione, todas las estaciones se deben de configurar con la misma velocidad. Cada computadora se conecta a un concentrador llamado MAU (Media Access Unit), y aunque la red queda físicamente en forma de estrella, lógicamente funciona en forma de anillo por el cual da vueltas el Token.

En realidad es el MAU el que contiene internamente el anillo y si falla una conexión automáticamente la ignora para mantener cerrado el anillo.

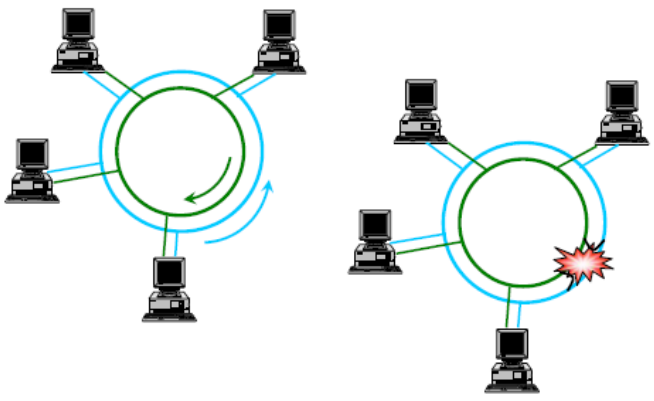


Fig. 1. 2. Modelo de una red Token Ring [2]

[2] http://www.diatel.upm.es/oortiz/Transporte%20de%20Datos/Teoria/2.5-Redes%20de%20area%20local.pdf

1.1.1.1.1.2 Redes Ethernet

Ethernet hace referencia a la familia de red de área local (LAN) del estándar IEEE 802.3, estándar que define lo que comúnmente se conoce como CSMA/CD (acceso múltiple con detección de portadora y detección de colisiones).

Otras tecnologías y protocolos han sido promocionados como probables sustitutos para Ethernet, sin embargo, Ethernet ha sobrevivido como principal tecnología de red local, ya que su protocolo tiene las siguientes características:

- Fácil de entender, implementar, administrar y mantener.
- Bajo costo para la implementación de la red.
- Flexibilidad.
- Garantía de conexión.

Varias son las topologías en las que Ethernet funciona, inicialmente se manejaba una estructura de cable coaxial en forma de bus. La longitud del segmento se limita a 500 mts y abarca hasta 100 estaciones conectadas a un solo segmento.

Fig. 1. 3. Diagrama de interconexión bus [3]

Desde principios de los 90, la configuración y el diseño de las redes se inclinan por la topología de estrella. La unidad central de la red es o un repetidor multipuertos o un conmutador de red. Todas las conexiones de red en una estrella son de punto a punto.

[3] http://www.pcdoctor.com.mx/Radio%20Formula/temas/Redes.htm

Fig. 1. 4. Diagrama de Ethernet Estrella [4]

En la topología de anillo cada estación está conectada a la siguiente y la última se conecta a la primera, de esta manera se forma un anillo de comunicaciones. Cada estación tiene un receptor y un transmisor que hace la función de repetidor, pasando la señal a la siguiente estación del anillo, de esta manera se evitan las colisiones. Cabe mencionar que si algún nodo de la red cae o deja de funcionar, la comunicación en todo el anillo se pierde. En un anillo doble, los datos no se pierden ya que estos se envía en ambas direcciones pero creando redundancia (tolerancia a fallos).

Fig. 1. 5. Diagrama de Ethernet Anillo [5]

La topología en árbol es parecida a varias redes en estrella interconectadas sin un nodo central. El flujo de información es jerárquico y para transmitir se realiza un control de acceso al medio.

[4] http://www.pcdoctor.com.mx/Radio%20Formula/temas/Redes.htm
[5] http://www.pcdoctor.com.mx/Radio%20Formula/temas/Redes.htm

Fig. 1. 6. Diagrama de Ethernet Árbol [6]

La topología de malla se implementa para proporcionar la mayor protección posible para evitar una interrupción del servicio.

Fig. 1. 7. Diagrama de Ethernet Malla [7]

1.1.1.1.1.3 Redes VLAN

Uno de los problemas que se puede encontrar en una red es la confidencialidad entre usuarios, también, el desperdicio de ancho de banda. La solución a este problema fue la división de la LAN en segmentos lógicos, los cuales son independientes entre sí, pero esto provoca la imposibilidad entre las LAN para comunicarse con algunos de los usuarios de la misma.

[6] http://www.pcdoctor.com.mx/Radio%20Formula/temas/Redes.htm
[7] http://www.pcdoctor.com.mx/Radio%20Formula/temas/Redes.htm

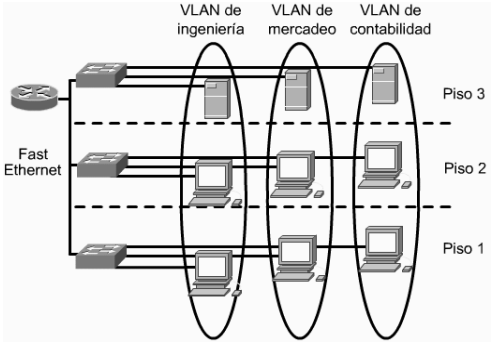


Fig. 1. 8. Modelo de una red VLAN [8]

La tecnología de las VLAN se basa en el empleo de switchs, esto permite un control más inteligente del tráfico de la red, ya que este dispositivo trabaja a nivel de la capa 2 del modelo OSI y es capaz de aislar el tráfico, incrementando la eficiencia de la red.

Por otro lado, al distribuir a los usuarios de un mismo grupo lógico a través de diferentes segmentos, se logra el incremento del ancho de banda en dicho grupo de usuarios.

Tipos de VLAN:

- **VLAN de puerto central:** Es en la que todos los nodos de una VLAN se conectan al mismo puerto del switch.
- **VLAN Estáticas:** Los puertos del switch están pre asignados a las estaciones de trabajo.
- **Por puerto:** Se indica qué puertos pertenecen a cada VLAN.
- **Por dirección MAC:** Los miembros de la VLAN están especificados en una tabla por su dirección MAC.
- **Por protocolo:** Asigna a un protocolo una VLAN.
- **Por direcciones IP:** Está basado en el encabezado de la capa 3 del modelo OSI. No actúa como router sino para hacer un mapeo de que direcciones IP están autorizadas a entrar en la red VLAN.

[8] http://www.eduangi.com/documentos/3_CCNA2.pdf

- **Por nombre de usuario:** Se basan en la autenticación del usuario y no por las direcciones MAC de los dispositivos.
- **VLAN Dinámicas (DVLAN):** Los puertos del switch automáticamente determinan a que VLAN pertenece cada puesto de trabajo, basándose en direcciones MAC, direcciones lógicas o protocolos utilizados.

1.1.1.1.1.4 Redes WLAN

WLAN es un sistema de comunicación de datos inalámbrico flexible, muy utilizado como alternativa a las tradicionales redes LAN. Usa tecnología de radiofrecuencia que permite mayor movilidad a los usuarios.

Características:

- **Movilidad:** Permite transmitir información en tiempo real en cualquier lugar y a cualquier usuario. Esto implica mayor productividad.
- **Facilidad de instalación:** Al no utilizar cables, se minimiza el tiempo de instalación.
- **Flexibilidad:** Puede llegar donde el cable no puede, superando obstáculos. Así, es útil en zonas donde el cableado no es posible o es muy costoso.

Existen dos tipos de redes inalámbricas: la Ad-Hoc[9] y la Infraestructure.[10]

- **Ad-Hoc:** Es una conexión de tipo punto a punto, en la que los clientes se conectan directamente unos con otros, cada nodo forma parte de una red Peer to Peer, aquí, se envían los paquetes de información al aire, con la esperanza que lleguen a su destino.

[9] Ad-hoc. Conexión de tipo punto a punto, en la que los clientes se conectan directamente unos con otros, aquí, se envían los paquetes de información al aire, con la esperanza que lleguen a su destino.
[10] Infraestructure. Redes inalámbricas que se conectan mediante puntos de acceso.

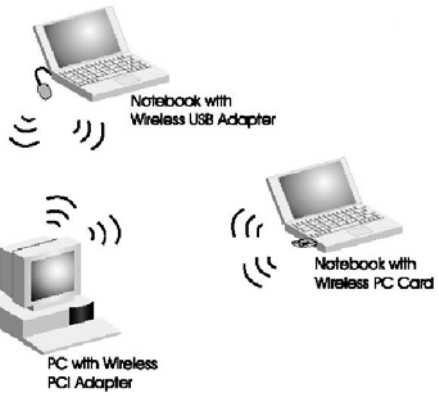


Fig. 1. 9. Modelo de una red Ad-hoc [11]

- **Infraestructure:** Se utiliza un dispositivo llamado punto de acceso, que funciona como el HUB tradicional. Envía directamente los paquetes de información a cada computador de la red. Este modo puede soportar hasta un máximo de 2048 usuarios.

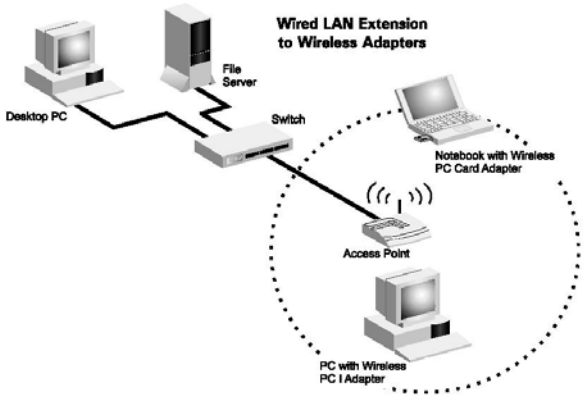


Fig. 1. 10. Modelo de una red Infraestructure [12]

La seguridad es un factor importante, ya que al utilizar ondas de radio implica un riesgo sobre las redes sin cables, ya que la señal puede ser recogida por cualquier receptor. Normalmente se utilizan sistemas de encriptación para reforzar la seguridad en las redes inalámbricas. El más utilizado es el WEP (Wired Equivalent Privacy) que utiliza encriptación de hasta 512 bits.

[11] http://lwwa175.servidoresdns.net:9000/proyectos_wireless/Web/topologias.htm
[12] http://lwwa175.servidoresdns.net:9000/proyectos_wireless/Web/topologias.htm

Componentes de una red WLAN:

- **Cliente**: Cada computador que acceda a la red se denomina cliente.
- **Punto de Acceso**: Hace las veces de un concentrador. Envía los paquetes de información directamente al computador indicado, esto mejora la velocidad y eficiencia de la red.
- **Antena**: Amplifican la señal. Las antenas direccionales emiten en una sola dirección y es preciso orientarlas a mano. Las antenas omnidireccionales emiten y reciben señal en 360º.

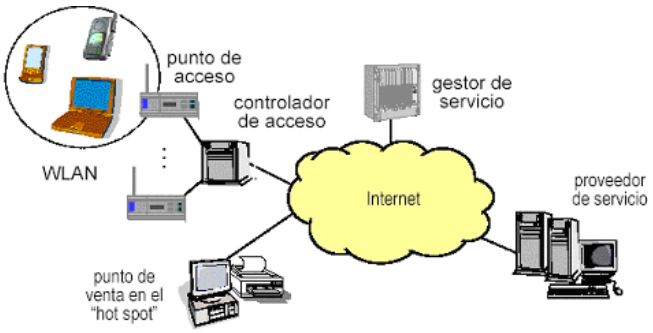


Fig. 1. 11. Esquema de una red WLAN [13]

Un único punto de acceso puede soportar un pequeño grupo de usuarios y puede funcionar en un rango de al menos treinta metros y hasta varios cientos.

El punto de acceso es normalmente colocado en alto pero podría colocarse en cualquier lugar en que se obtenga la cobertura de radio deseada. El usuario final accede a la red WLAN a través de adaptadores inalámbricos.

1.1.1.2 Redes de área metropolitana (MAN)

Otro tipo de red que se emplea en las organizaciones es la red de área metropolitana o MAN (Metropolitan Area Network), una versión más grande que la LAN y que normalmente se basa en una tecnología similar a ésta.

[13] http://www.unavarra.es/organiza/etsiit/cas/estudiantes/pfc/redaccna/Tecnologias%20de%20Acceso/WLAN/elementos%20red/esquema_red.htm

Una red de área metropolitana es una red de alta velocidad (banda ancha) que dando cobertura en un área geográfica extensa, proporciona capacidad de integración de múltiples servicios mediante la transmisión de datos, voz y vídeo, sobre medios de transmisión tales como fibra óptica y par trenzado de cobre a velocidades que normalmente van desde los 2 Mbits/s hasta 155 Mbits/s.

Los componentes de una red de área metropolitana son:

- **Puestos de trabajo:** Son los sistemas desde los cuales el usuario demanda las aplicaciones y servicios proporcionados por la red.
- **Nodos de red:** Son dispositivos que proporcionan servicio a los puestos de trabajo que forman parte de la red. Sus funciones son: almacenamiento, filtrado y conversión de información.
- **Sistema de cableado:** Está constituido por el cable utilizado para conectar entre sí los nodos de red y los puestos de trabajo.
- **Protocolos de comunicación:** Reglas y procedimientos para establecer la comunicación de los nodos.
- **Aplicaciones:** Como Sistemas de Tratamiento de Mensajes (MHS), Gestión, Acceso y Transferencia de Ficheros (FTAM) puede ser las posibles aplicaciones que soporte la red.

Las redes MAN adoptan un estándar (DQDB), que equivale a la norma IEEE 802.6. EL DQDB consiste en dos buses unidireccionales, los cuales se conectan a todas las computadoras.

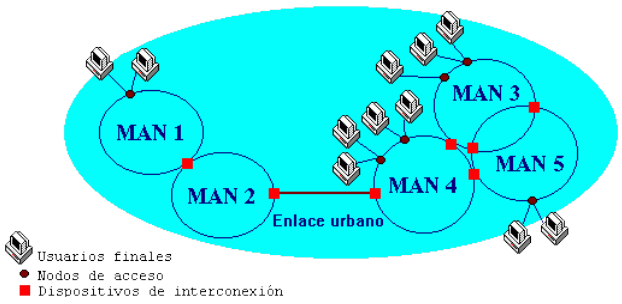


Fig. 1. 12. Esquema de una red MAN [14]

[14] http://www.csi.map.es/csi/silice/Redman4.html

1.1.1.3 Redes de área amplia (WAN)

La Red WAN (Wide Area Netowork) es una red de comunicación de datos que tiene una cobertura geográfica grande y utiliza como medio de transmisión el que ofrecen las operadoras de servicios de telefonía.

Las Redes WAN pueden establecer comunicación con:

- **Enlaces punto a punto:** Se les conoce como líneas privadas, ya que su trayectoria es permanente y fija.
- **Conmutación de circuitos:** Es un método de conmutación en el que se establece, mantiene y termina un circuito físico dedicado a través de una red de transporte para cada sesión de comunicación.
- **Conmutación de paquetes:** Los dispositivos conectados a la red comparten un solo enlace para transferir los paquetes desde el origen al destino. Las redes *Frame Relay, ATM y X.25* son ejemplo de estas.
- **Circuitos virtuales WAN:** Es un circuito lógico creado para asegurar una comunicación confiable entre dos dispositivos de red. Hay dos tipos de circuitos virtuales, los virtuales conmutados y los virtuales permanentes.

Entre las WAN mas grandes se encuentran: ARPANET, creada por la Secretaría de Defensa de los Estados Unidos en la década de los 60 y que se convirtió en lo que actualmente es la WAN mundial, Internet.

1.1.1.3.1 Componentes de una red WAN

Una red de área amplia puede ser descrita como un grupo de redes individuales conectadas a través de extensas distancias geográficas y que utilizan tecnologías de transporte de alta velocidad.

Los componentes de una red WAN típica incluyen:

- Dos o más redes LAN independientes.
- Routers conectados a las LAN.
- Dispositivos de acceso al enlace.
- Enlaces de inter red WAN.

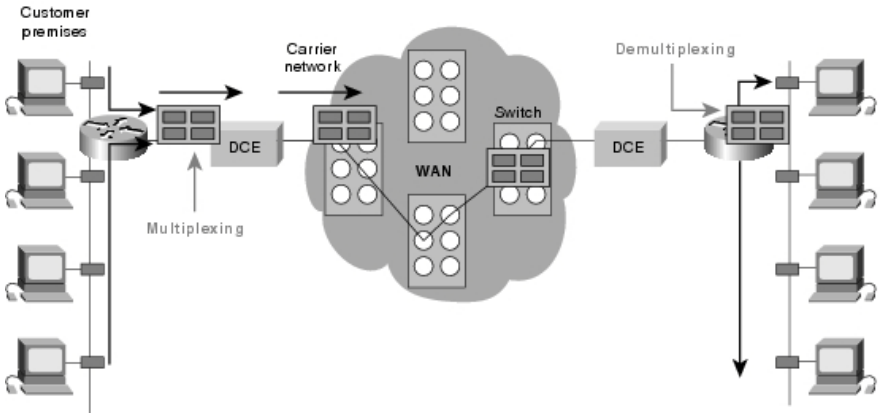


Fig. 1. 13. Esquema de una red WAN [15]

1.2 MODELO DE REDES

1.2.1 INTRODUCCIÓN

Por mucho tiempo se consideró al diseño de redes un proceso muy complicado de llevar a cabo, debido a que los fabricantes de computadoras tenían su propia arquitectura de red, y en ningún caso existía compatibilidad entre marcas. Posteriormente la ISO (Organización Internacional de Normalización) en 1977 desarrolla una estructura de normas comunes dentro de las redes.

La idea fue diseñar redes como una secuencia de capas, cada una construida sobre la anterior.

[15] http://www.cisco.com/en/US/docs/internetworking/technology/handbook/Intro-to-WAN.html#wp1020550

Las capas se pueden dividir en dos grupos:

- Servicios de transporte (niveles 1, 2, 3 y 4)
- Servicios de soporte al usuario (niveles 5, 6 y 7)

OSI nace de una necesidad de normalizar los elementos que participan en la solución de los problemas de comunicación entre equipos de diferentes fabricantes.

1.2.2 ESTRUCTURA DEL MODELO OSI DE ISO

- **Estructura multinivel:** Se diseña una estructura multinivel con la idea de que cada nivel resuelva solo una parte del problema de la comunicación.
- **El nivel superior utiliza los servicios de los niveles inferiores:** La comunicación entre niveles se define de manera que un nivel N utilice los servicios del nivel N-1 y proporcione servicios al nivel N+1.
- **Puntos de acceso:** Entre los diferentes niveles existen interfaces llamadas *Puntos de Acceso* a los servicios.
- **Dependencia de Niveles:** Cada nivel es dependiente del nivel inferior como así también lo es del nivel superior.
- **Encabezados:** En cada nivel, se incorpora al mensaje un formato de control. Este elemento de control permite que un nivel en la computadora receptora se entere de que la computadora emisora le está enviando un mensaje con información.

Cualquier nivel puede incorporar un encabezado al mensaje, por esta razón se considera que un mensaje está constituido de dos partes, el encabezado y la información. Entonces, la incorporación de encabezados es necesaria aunque represente un lote extra en la información. Sin embargo, como la computadora receptora retira los encabezados en orden inverso a como se enviaron desde la computadora emisora, el mensaje original no se afecta.

1.2.2.1 Capas del modelo OSI

Capa Física

Aquí se encuentran los medios materiales para la comunicación como las placas, cables, conectores, es decir los medios mecánicos y eléctricos. La capa física se ocupa de la transmisión de bits a lo largo de un canal de comunicación, la misma debe garantizar que un bit que se envía desde un nodo origen llegue con el mismo valor al nodo final.

Capa de Enlace

Se encarga de transformar la línea de transmisión común en una línea sin errores para la capa de red, esto se lleva a cabo dividiendo la entrada de datos en tramas, por otro lado se incluye un patrón de bits entre las tramas de datos. Esta capa también se encarga de solucionar los problemas de reenvío, o mensajes duplicados cuando hay destrucción de tramas. Por otro lado es necesario controlar el tráfico de datos. El nivel de enlace trata de detectar y corregir los errores.

Servicios para el nivel de red:

- **Servicio sin acuses de recibo.** Es apropiado si la frecuencia de errores es muy baja o el tráfico es de tiempo real (por ejemplo, voz).
- **Servicio con acuses de recibo.** El receptor envía un acuse de recibo al remitente para cada trama recibida.

Control de flujo

Se usan técnicas de control de flujo para no saturar al receptor de un emisor. El control de flujo involucra dos acciones: detección y corrección de errores. Estas acciones son altamente importantes en el funcionamiento de una red.

Detección y corrección de errores

HDLC (High-level Data Link Control). Este es un protocolo orientado a bit, es decir, sus especificaciones cubren qué información lleva cada uno de los bits de la trama.

Fig. 1. 14. Esquema de la Trama HDLC[16]

Como se puede ver en la figura, se definen unos campos que se agregan a la información (datos). Estos campos se utilizan con distintos fines. Con el campo Checksum se detectan posibles errores en la transmisión mientras que con el campo control se envía mensajes como datos recibidos correctamente, etc.

Capa de Red

Este nivel encamina los paquetes de la fuente al destino final a través de routers intermedios. Tiene que saber la topología de la subred, evitar la congestión, y manejar saltos cuando la fuente y el destino están en redes distintas.

Otro problema a solucionar por esta capa es la interconexión de redes heterogéneas, solucionando problemas de protocolos diferentes, o direcciones desiguales.

Capa de Transporte

La función principal es de aceptar los datos de la capa superior y dividirlos en unidades más pequeñas, para pasarlos a la capa de red, asegurando que todos los segmentos lleguen correctamente, esto debe ser independiente del hardware

[16] http://www.eie.fceia.unr.edu.ar/~etapia/Redes2002pub/X25.pdf

en el que se encuentre. Para bajar los costos de transporte se puede multiplexar varias conexiones en la misma red.

Las funciones del nivel de transporte pueden ser independientes de las funciones del nivel de red. Las aplicaciones pueden usar estas funciones para funcionar en cualquier tipo de red.

Protocolos de transporte

Los protocolos de transporte se parecen a los protocolos de enlace. Ambos manejan el control de errores, el control de flujo, la secuencia de paquetes, etc.; sin embargo, existen diferencias, pues en el nivel de transporte, se necesita una manera para especificar la dirección del destino.

Existen dos protocolos importantes operando sobre esta capa: TCP y UDP.

Capa de Sesión

Permite a los usuarios sesionar entre sí permitiendo acceder a un sistema de tiempo compartido a distancia, o transferir un archivo entre dos máquinas. Uno de los servicios de esta capa es la del seguimiento de turnos en el tráfico de información, como así también la administración de tareas, sobre todo para los protocolos.

Otra tarea de esta capa es la de sincronización de operaciones con los tiempos de caída en la red.

Capa de Presentación

Se ocupa de los aspectos de sintaxis y semántica de la información que se transmite, por ejemplo la codificación de datos según un acuerdo.

Esto se debe a que los distintos formatos en que se representa la información que se transmite son distintos en cada máquina. Otro aspecto de esta capa es la compresión de información reduciendo el número de bits.

Capa de Aplicación

El nivel de aplicación es siempre el más cercano al usuario. Por nivel de aplicación se entiende el programa o conjunto de programas que generan una información para que esta viaje por la red.

El ejemplo más claro sería el del correo electrónico y transferencias de archivos; cuando se procesa y se envía un correo electrónico este puede ir en principio a cualquier lugar del mundo, y ser leído en cualquier tipo de computador.

Fig. 1. 15. Arquitectura de la red basada en el modelo OSI [17]

1.2.3 EL MODELO POR CAPAS DE TCP/IP DE INTERNET

El segundo modelo por capas de mayor diseño no se origina de un comité de estándares, sino que procede de las investigaciones que se realizan respecto al conjunto de protocolos de TCP/IP.

[17] http://neo.lcc.uma.es/evirtual/cdd/tutorial/modelos/Mtcp.html

El modelo TCP/IP está organizado en cuatro capas conceptuales que se construyen sobre una quinta capa de hardware, como se presenta en la Fig. 1.16.

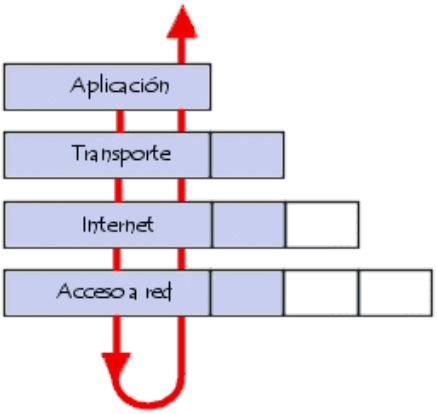


Fig. 1. 16. Modelo de capas TCP [18]

Capa de interfaz de red

La capa de interfaz es responsable de aceptar los datagramas IP y transmitirlos hacia una red específica. Una interfaz de red puede consistir en un dispositivo controlador o un complejo subsistema que utiliza un protocolo de enlace de datos propios (cuando la red consta de conmutadores de paquetes que se comunican con anfitriones utilizando HDLC).

Capa Internet

La capa Internet maneja la comunicación de una máquina a otra, la entrada de datagramas, verifica su validez y utiliza un algoritmo de ruteo para decidir si el datagrama debe procesarse de manera local o debe ser transmitido. Por último, la capa Internet envía los mensajes ICMP de error y control necesarios y maneja todos los mensajes ICMP entrantes.

[18] http://es.kioskea.net/contents/internet/tcpip.php3

Capa de transporte

Proporciona la comunicación entre una aplicación y otra, este tipo de comunicación se conoce como comunicación punto a punto.

La capa transporte regula el flujo de información, proporciona un transporte confiable, asegurando que los datos lleguen sin errores y en secuencia. El software de transporte divide el flujo de datos que se está enviando en pequeños fragmentos (paquetes) y pasa cada paquete, con una dirección de destino, hacia la siguiente capa de transmisión.

Un computador puede tener varios programas de aplicación accediendo a la red al mismo tiempo. La capa de transporte debe aceptar datos desde varios programas de usuario y enviarlos a la capa del siguiente nivel. Para hacer esto, se añade información adicional a cada paquete, incluyendo códigos que identifican qué programa de aplicación envía y qué programa debe recibir.

Capa de aplicación

Es el nivel más alto, una aplicación interactúa con uno de los protocolos de nivel de transporte para enviar o recibir datos. Cada programa de aplicación selecciona el tipo de transporte necesario. El programa de aplicación pasa los datos en la forma requerida hacia el nivel de transporte para su entrega.

1.2.4 DIFERENCIAS Y SIMILITUDES ENTRE OSI y TCP

Similitudes entre el modelo OSI y el modelo TCP/IP

- Ambos se dividen en capas o niveles.
- Utilizan tecnología de conmutación de paquetes.
- Ambos manejan una fase de establecimiento de conexión, fase de transferencia y finalmente una fase de liberación de la conexión.

Diferencias entre el modelo OSI y el modelo TCP/IP

- OSI distingue de forma clara los servicios, las interfaces y los protocolos. TCP/IP no lo hace así, no dejando de forma clara esta separación.
- OSI fue definido antes de implementar los protocolos, por lo que algunas funcionalidades necesarias fallan o no existen. En cambio, TCP/IP se creó después que los protocolos, por lo que se amolda a ellos perfectamente.
- TCP/IP parece ser más simple porque tiene menos capas.

1.2.5 EL MODELO IEEE

Otro modelo de red fue desarrollado por el mismo instituto de Ingenieros Eléctricos y Electrónica (IEEE). Debido a la proliferación de Redes de Área Local (LAN) muchos productos aparecieron, entonces la IEEE empezó a definir estándares de red. El proyecto fue llamado 802, por el año y el mes en que empezó: Febrero de 1980.

Del proyecto 802 resultaron numerosos documentos, incluyendo los tres principales estándares para topologías de red.

1.2.5.1 IEEE 802.3

La especificación IEEE para Ethernet es la 802.3, define que tipo de cableado se permite y cuáles son las características de la señal que transporta. La especificación 802.3 original utilizaba un cable coaxial grueso de 50 ohm, que permite transportar una señal de 10 Mbps a 500 m. Mas tarde se añadió la posibilidad de utilizar otros tipos de cables: coaxial delgado; pares de cables trenzados, y fibra óptica.

Una red Ethernet puede transmitir datos a 10 Mbps sobre un solo canal de banda base, generalmente un bus coaxial o una estructura ramificada. Los segmentos de cable están limitados a un máximo de 500 m. Aunque la mayoría de

fabricantes especifican un máximo de 100 estaciones en cada segmento, el límite práctico puede ser menor, dependiendo de la utilización.

Cuatro son los tipos de datos definidos para operar con cables UTP y fibra óptica:

- 10 Mbps - 10Base-T Ethernet (IEEE 802.3)
- 100 Mbps - Fast Ethernet (IEEE 802.3u)
- 1000 Mbps - Gigabit Ethernet (IEEE 802.3z)
- 10-Gigabit - 10 Gbps Ethernet (IEEE 802.3ae).

El formato de la trama es el siguiente:

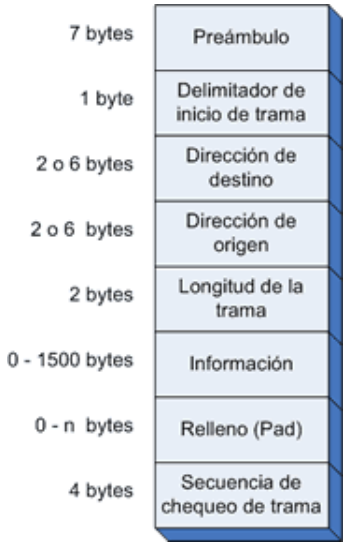


Fig. 1. 17. Formato de trama 802.3 [19]

Preámbulo: Este campo tiene una extensión de 7 bytes que siguen la secuencia "10101010", semejante a la de bandera señalizadora del protocolo HDLC.

Inicio: Es un campo de 1 byte con la secuencia "10101011" que indica que comienza la trama.

[19] http://www.textoscientificos.com/redes/ethernet/control-acceso-medio-csma-cd

Dirección de destino: Es un campo de 2 o 6 bytes, el bit de mayor orden de este campo, que ocupa el lugar 47, codifica si la dirección de destino es un único destinatario (bit puesto a 0) o si representa una dirección de grupo (bit puesto a 1). Una dirección de grupo es la dirección a la que varias estaciones tienen derecho de escucha. Cuando todos los bits del campo dirección están a 1, se codifica una difusión o *broadcast*, es decir, codifica una trama para todas las estaciones de la red.

Dirección de origen: Codifica la dirección MAC de la tarjeta que originó la trama.

Longitud datos: Este campo de dos bytes codifica los bytes que contiene el campo de datos. Su valor oscila en un rango entre 0 y 1.500.

Datos: Es un campo que puede codificar entre 0 y 1.500 bytes.

Relleno: La IEEE 802.3 especifica que una trama no puede tener un tamaño inferior a 64 bytes, por tanto, cuando la longitud del campo para completar una trama mínima de, al menos, 64 bytes.

CRC: Se codifica el control de errores de la trama.

1.2.5.2 IEEE 802.5

Token Ring es un protocolo de red local, se define en el estándar IEEE 802.5, donde todas las estaciones se conectan en un anillo, y cada estación puede escuchar directamente las transmisiones sólo de su vecino inmediato. La autorización para transmitir se concede por un mensaje (token) que circula en todo el anillo.

Las redes Token Ring funcionan básicamente de la siguiente forma: si ningún host desea transmitir, todos están en modo escucha y el token va pasando de un host a otro indefinidamente. Cuando algún nodo desea transmitir debe esperar a que pase por él el token. En ese momento, se apodera de éste, pasa a modo

transmisión y envía la trama al siguiente nodo. Todos los demás hosts del anillo, incluido el destino, siguen en modo escucha, retransmitiendo la trama hacia el siguiente nodo. El host destino, además de retransmitirlo, retiene una copia del trama.

El formato de la trama es el siguiente:

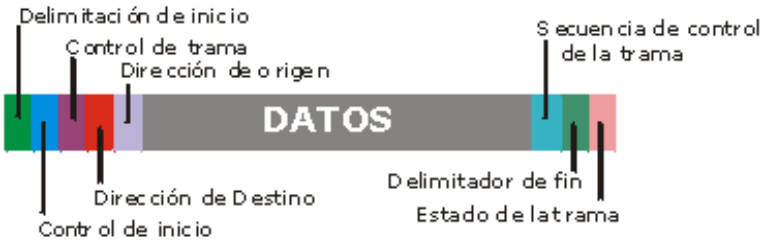


Fig. 1. 18. Formato de trama 802.5 [20]

Delimitador de Comienzo (DC): 1 Byte que marcar el inicio de la trama.

Control de Acceso: 1 Byte que contiene bits especiales: tres de prioridad, el del token, el de monitor y tres de reserva. Su forma es PPPTMRRR.

Dirección Destino: 6 Bytes que indican la dirección del nodo destino.

Dirección Origen: 6 Bytes que indican la dirección del nodo fuente.

Datos: Campo que encapsula los datos del nivel superior, limitado en tamaño por el Token Holding Time.

CRC: 4 Bytes que corresponden a una suma de verificación para asegurar que el frame llegó en buen estado.

Delimitador de Fin de trama (DF): 1 Byte que marca el final de la trama.

[20] http://mx.geocities.com/salazarojm/ring.htm

Estado del Frame: 1 Byte que contiene dos bits especiales, el A y el C. Al llegar un frame al destino, éste coloca el bit A en uno y si el nodo copia el frame coloca el bit C en uno. Con esto, el nodo emisor al recibir su frame tiene las siguientes opciones en los bits AC: 00 destino no presente, 10 destino presente y frame no aceptado, 11 destino presente y frame copiado. Con esto, el protocolo incorpora un mecanismo automático de acuse de recibo.

1.3 ARQUITECTURA DE REDES

La arquitectura de red es el medio más efectivo para desarrollar e implementar un conjunto de productos que se puedan interconectar. La arquitectura es el plan a ejecutarse para conectar los protocolos y otros programas de software. Esto es provechoso tanto para los usuarios de la red como para los proveedores de hardware y software.

Las redes se diseñan en capas con el propósito de reducir la complejidad, pero la cantidad de capas, las funciones que se llevan a cabo en cada una y el nombre varían de una red a otra. Cada una de las capas libera a la posterior del conocimiento de las funciones subyacentes. Esto requiere establecer interfaces de comunicación entre capas que definen los servicios y operaciones.

Cuando una capa-n de una máquina A establece comunicación con la capa-n una máquina B, se establecen reglas y convenciones para llevarla a cabo, lo cual se denomina protocolo de la capa-n. A la configuración de capas y protocolo se le denomina arquitectura de red.

Características de la arquitectura de redes:

- **Separación de funciones:** Mediante la arquitectura de red un sistema se diseña con alto grado de modularidad, de manera que se puedan hacer cambios con un mínimo de perturbaciones.
- **Amplia conectividad:** El objetivo de las redes es proveer conexión óptima entre cualquiera de sus nodos.

- **Recursos compartidos:** Mediante las arquitecturas de red se pueden compartir recursos como impresoras y bases de datos, y a su vez se consigue que la operación de la red sea más eficiente y económica.
- **Administración de la red:** Dentro de la arquitectura se debe permitir que el usuario defina, opere, cambie, proteja y de mantenimiento a la red.
- **Facilidad de uso:** Los diseñadores desarrollan interfaces gráficas haciéndolas amigables para el usuario.
- **Normalización:** Mientras mayor es la normalización, mayor es la colectividad y menor el costo.
- **Administración de datos:** En las arquitecturas de red se toma en cuenta la administración de los datos y la necesidad de interconectar los diferentes sistemas de administración de bases de datos.
- **Interfaces:** La arquitectura de red combina los protocolos apropiados (los cuales se escriben con programas de computadora) y paquetes de software para producir una red funcional.
- **Aplicaciones**: En las arquitecturas de red se separan las funciones que se requieren para operar una red a partir de las aplicaciones comerciales de la organización. Se obtiene más eficiencia cuando los programadores del negocio no necesitan considerar la operación.

1.3.1 ARQUITECTURA DE RED BASADA EN EL MODELO OSI

El modelo OSI surge como una búsqueda de solución al problema de incompatibilidad de las redes de los años 60. Fue desarrollado por la ISO (International Organization for Standardization) en 1977 y adoptado por ITU-T.

Consiste de una serie de niveles que contienen normas funcionales que cada nodo de red debe seguir para el intercambio de información y la inter operatividad de los sistemas.

Fig. 1. 19. Modelo de referencia OSI [21]

Cada nivel del OSI es un modulo independiente que provee un servicio para el nivel superior dentro de la Arquitectura o modelo.

Esta arquitectura utiliza la terminología de las primeras redes llamadas ARPANET, donde las maquinas que se utilizan para correr los programas en la red se llaman computadoras centrales, o también llamadas terminales.

Los terminales se comunican a través de una subred de comunicaciones que se encarga de enviar los mensajes entre los terminales, como si fuera un sistema de comunicación telefónica.

La subred se compone de dos elementos: las líneas de transmisión de datos, y los elementos de conmutación, llamados IMP (procesadores de intercambio de mensajes). De ésta manera todo el tráfico que va o viene a un terminal pasa a través de su IMP.

El diseño de una subred puede ser de dos tipos:

- Canales punto a punto
- Canales de difusión

[21] http://www.cisco.com/en/US/docs/internetworking/technology/handbook/Intro-to-WAN.html#wp1020550

El primero (punto a punto) contiene varias líneas de comunicaciones, conectadas cada una a un par de IMP. Cuando un mensaje (paquete) se envía de un IMP otro, se utiliza un IMP intermedio, que garantiza el envío del mensaje, esta modalidad se utiliza en las redes extendidas que son del tipo almacenamiento y reenvío.

El segundo (difusión) contiene un solo canal de difusión que se comparte con todas las máquinas de la red. Los paquetes que una máquina quiera enviar, son recibidos por todas las demás, un campo de dirección indica quién es el destinatario, este modelo se utiliza en redes locales.

1.3.2 ARQUITECTURA DE RED X.25

La norma X.25 es un estándar recomendado por la CCITT para redes de paquetes. X.25 es una norma de interfaz de gran cobertura, aunque no es muy rápida ofrece un servicio orientado a conexión, es fiable, y ofrece multiplexación.

Para que las redes de paquetes y los equipos terminales se logren interconectar es necesario disponer de mecanismos de control, siendo el más importante, el control de flujo, que evita la congestión de la red. El DTE controla el flujo de datos proveniente de la red.

Además deben existir procedimientos de control de errores para garantizar la correcta recepción de todo el tráfico. X.25 proporciona estas funciones de control de flujo y de errores.

Las sesiones X.25 se realizan cuando un dispositivo DTE contacta a otro para solicitar el establecimiento de una sesión. El dispositivo DTE que recibe la solicitud está en la capacidad de aceptar o rechazar la conexión. Si se acepta la solicitud, los dos sistemas comienzan con la transferencia de información en modo full-duplex. Después de que la sesión se termina se requiere el establecimiento de una nueva sesión en caso de que se desee establecer otra comunicación.

El protocolo X.25 define tres niveles: Nivel Físico, Nivel de Tramas y Nivel de Paquetes, estos niveles definen las funciones de los niveles Física, de Enlace de Datos y de Red del Modelo OSI.

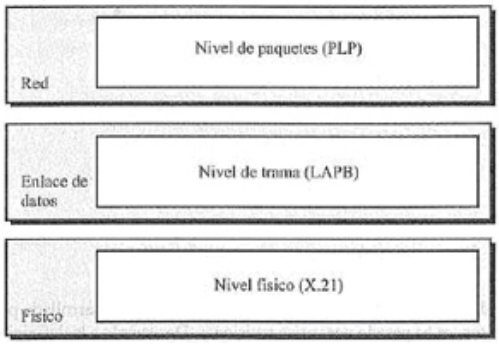


Fig. 1. 20. Arquitectura de red X.25 [22]

1.3.2.1 Características del protocolo X.25

X.25 trabaja sobre servicios basados en circuitos virtuales (CV) o canales lógicos en el cual el usuario (ETD) piensa que es un circuito dedicado a un sólo computador; pero la verdad es que lo comparte con varios usuarios (ETD) mediante técnicas de multiplexado entrelazando paquetes de distintos usuarios de un mismo canal lógico.

Es aconsejable utilizar de la norma X.25 porque:

- Adopta un estándar común para distintos fabricante, permite conectar equipos de marcas distintas.
- Empleando la norma X.25 se reduce los costos de la red.
- Es más sencillo solicitar a un fabricante una red adaptada a la norma X.25 que entregarle un extenso conjunto de especificaciones.

La función más importante que proporciona X.25 para que las redes de paquetes y estaciones de usuario se puedan interconectar es el Control de Flujo.

[22] http://www.mailxmail.com/curso-redes-protocolos-1/protocolos-estandares-introduccion-2-2

El Control de Flujo es importante porque:

- Evita la congestión de la red.
- Permite la recuperación de Errores.
- Proporciona identificación de paquetes procedentes de computadoras y terminales concretos.
- Permite confirmación y rechazo de paquetes.

X.25 no incluye algoritmos de enrutamiento, pero a pesar que los interfaces DTE/DCE son independientes en ambos extremos de la red, X.25 interviene desde un extremo hasta el otro, ya que el tráfico seleccionado es encaminado de principio a fin.

1.3.2.2 Capas de funcionalidad X.25

X.25 está formado por tres capas de funcionalidad, estas tres capas corresponden a las tres capas inferiores del modelo OSI.

Nivel Físico: La interfaz de nivel físico normaliza el diálogo entre el DCE y el DTE. Este nivel detalla los estándares para la transmisión y recepción de datos mecánica y eléctricamente.

Nivel de Enlace: Este nivel garantiza la comunicación y asegura la transmisión de datos entre dos equipos directamente conectados. En este nivel se utiliza el protocolo LAP-B[23] que forma parte del HDLC. Este protocolo define el fraccionamiento de los datos para la transmisión, y especifica la ruta que estos deben seguir a través de la red.

Nivel Red: Este nivel realiza detección y corrección de errores, compite por la retransmisión de los paquetes dañados.

[23] LAP-B (Link Access Procedure, Balanced) Protocolo de capa Enlace de Datos dentro del conjunto de protocolos de la norma X.25

1.3.2.3 Ventajas e inconvenientes de X.25

Como ventajas se puede citar:

- Varias conexiones lógicas sobre una física.
- Asignación dinámica de la capacidad (multiplexación estadística).
- Transporte de datos de múltiples sistemas.
- Fiable.

En cuanto a inconvenientes:

- Ancho de banda limitado.
- Retardo de transmisión grande y variable.
- Señalizaron en canal y común, ineficaz y problemática.

1.3.2.4 Limitaciones de la recomendación X.25

Esta arquitectura tiene las siguientes limitaciones:

- En cada nodo de la red se debe disponer de todo el paquete sin errores y se deben ejecutar los procesos de control y detección de errores.
- Los procesos en cada nodo son muy complejos.

Hoy en día, ya que la calidad de las redes es muy buena, se hace obsoleto el X.25 al ser tan rigurosa con los errores (esto implica coste y complejidad).

1.3.3 ARQUITECTURA SNA

La Arquitectura SNA[24] se refiere a una estructura integral que provee todos los modos de comunicación de datos y en la cual se puede planear e implementar nuevas redes de comunicación de datos.

[24] SNA: Systems Network Architecture.

La Arquitectura de Red SNA es un modelo de siete capas similar al Modelo OSI.

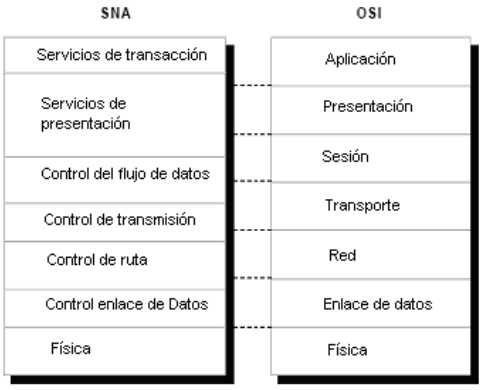


Fig. 1. 21. Arquitectura de red SNA[25]

La Arquitectura SNA se construyó tomando como base cuatro principios básicos:

- La Arquitectura SNA comprende funciones distribuidas, aquí, las responsabilidades de la red se puede trasladar de la computadora central a otros componentes de la red como son los equipos remotos.
- La Arquitectura SNA define trayectorias separadas entre usuarios finales, permitiendo realizar modificaciones a la configuración de la red sin afectar al usuario final.
- La Arquitectura SNA utiliza el principio de independencia de dispositivo, permitiendo añadir o modificar aplicaciones y equipos de comunicación sin afectar a otros elementos de la red.
- La Arquitectura SNA utiliza funciones y protocolos lógicos y físicos normalizados para la comunicación de información entre dos puntos.

Una red de comunicación de datos basada en los conceptos de Arquitectura SNA consta de lo siguiente:

- Computador principal.
- Procesador de comunicación de entrada (nodo intermedio).
- Controlador remoto inteligente (nodo intermedio o nodo de frontera).
- Diversas terminales de propósito general.

[25] http://upload.wikimedia.org/wikipedia/commons/8/87/SNAvsOSI_v1.png

1.3.4 ARQUITECTURA DE RED DIGITAL (DNA)

DNA[26] es una arquitectura de red distribuida implementada por la Digital Equipment Corporation. Se le llama DECnet y se compone de cinco capas. Las capas: Física, Enlace de Datos, Transporte y de Servicios de Red, estas corresponden casi exactamente a las cuatro capas inferiores del modelo OSI. La quinta capa, la de Aplicación, es una mezcla de las capas de presentación y aplicación del modelo OSI. La DECnet no cuenta con una capa de sesión separada.

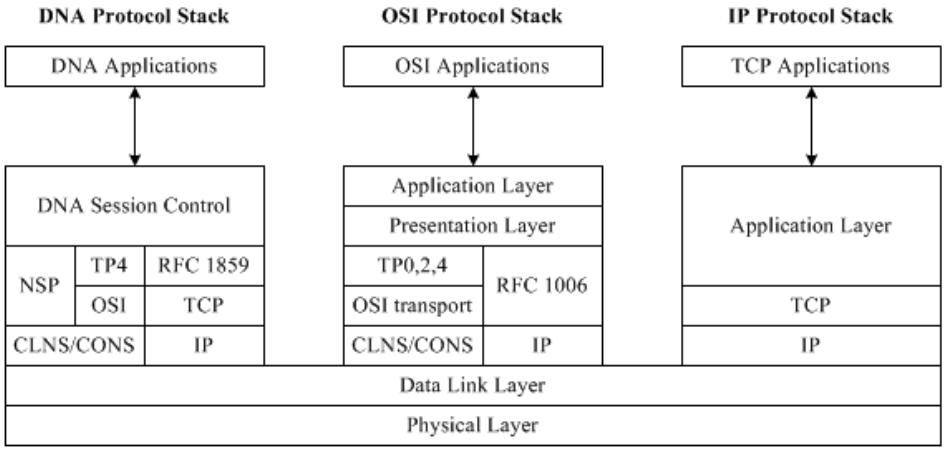


Fig. 1. 22. Arquitectura de red DNA [27]

El objetivo de la DECnet es permitir la interconexión de diferentes computadoras principales y redes punto a punto, multipunto o conmutadas de manera tal que los usuarios puedan compartir programas, archivos de datos y dispositivos de terminal remotos.

La DECnet soporta la norma del protocolo X.25 y cuenta con capacidades para conmutación de paquetes. El protocolo de mensaje para comunicación digital de datos (PMCDD) de la DECnet es un protocolo orientado a los bytes cuya estructura es similar a la del protocolo de Comunicación Binaria Síncrona (CBS) de IBM.

[26] DNA: Digital Network Architecture, también llamada DECnet (Digital Equipment Corporation)
[27] http://www.knowledgerush.com/kr/encyclopedia/DECNET/

1.3.5 ARQUITECTURA ARCNET

La Red de computación de recursos conectadas (ARCNET, Attached Resource Computing Network) es un sistema de red de banda base, con paso de testigo (token) que ofrece topologías en estrella y bus a un precio bajo. Las velocidades de transmisión son de 2.5 Mbits/seg.

ARCNET usa un protocolo de paso de testigo en una topología de red en bus con testigo, pero ARCNET no es considerada una norma IEEE. ARCNET es adecuada para entornos de oficina que usan aplicaciones basadas en texto y donde los usuarios no acceden frecuentemente a servidores. Las nuevas versiones de ARCNET soportan fibra óptica y par-trenzado. ARCNET es una buena elección cuando la velocidad no es un factor determinante pero el precio sí.

1.3.6 ARQUITECTURA ETHERNET

En 1972, Xerox Corporation creó el experimental Ethernet, y en 1975 introdujo el primer producto Ethernet. El Ethernet de Xerox fue tan exitoso que Xerox, Intel y Digital crearon un estándar para Ethernet de 10mbps. Este diseño fue la base de la especificación IEEE 802.3.

1.3.6.1 Funciones de la arquitectura Ethernet

Encapsulación de datos

- Formación de la trama estableciendo la delimitación correspondiente.
- Direccionamiento del nodo fuente y destino.
- Detección de errores en el canal de transmisión.

Manejo de Enlace

- Asignación de canal.
- Resolución de contención, manejando colisiones.

Codificación de los Datos

- Generación y extracción del preámbulo para fines de sincronización.
- Codificación y decodificación de bits.

Acceso al Canal

- Transmisión / Recepción de los bits codificados.
- Sensibilidad de portadora, indicando tráfico sobre el canal.
- Detección de colisiones, indicando contención sobre el canal.

1.4 CONVERGENCIA EN REDES DE DATOS

Una red convergente no es únicamente una red capaz de transmitir datos, video y voz en forma simultánea, sino un entorno en el que existen además servicios avanzados que integran estas capacidades, reforzando la utilidad de los mismos.

Existen varias definiciones del término convergencia, aunque habitualmente suele resumirse como:

- Capacidad de diferentes plataformas de red para transportar tipos de servicios básicamente similares.
- La aproximación de dispositivos de consumo tales como el teléfono, la televisión y el computador personal.

Esta última definición de convergencia es la que más se maneja, en la actualidad, los operadores de telecomunicaciones ofrecen programación audiovisual a través de sus redes y a la vez son importantes suministradores de acceso a Internet.

Los operadores tradicionales de redes de cable ahora prestan servicios integrados de telecomunicación, que incluye la telefonía vocal, acceso a Internet y sin duda la actividad tradicional de distribución de programas de televisión.

La convergencia está mostrándose ya como un motor fundamental de la actual evolución de las industrias de telecomunicaciones, medios de comunicación y tecnología de la información.

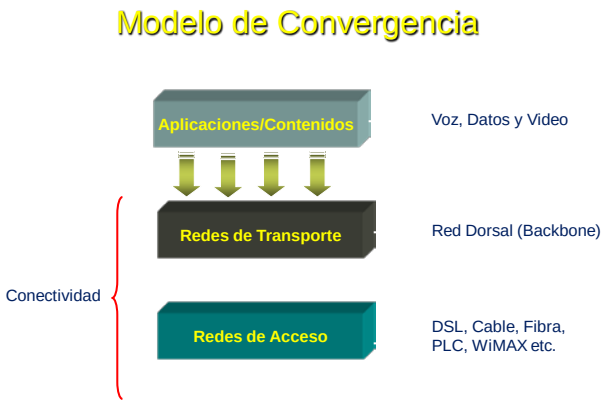


Fig. 1. 23. Modelo de Convergencia [28]

1.4.1 CONVERGENCIA DIGITAL

El considerable aumento de abonados de banda ancha, se debe a la gran demanda de mayor velocidad en Internet. Los servicios de banda ancha facilitan conexiones a Internet por lo menos cinco veces más rápidas que las típicas conexiones telefónicas, permitiendo a los usuarios jugar en línea y descargar música y vídeos, además de compartir archivos y acceder a la información con mayor rapidez y eficacia que antes.

1.4.2 CONVERGENCIA DE REDES Y SERVICIOS

Ante la revolución de la convergencia de redes y servicios, somos testigos de los cambios acelerados en el mundo de las telecomunicaciones.

El modelo de negocio tradicional de los proveedores de servicios de comunicación está cambiando rápidamente, de tal manera que estos proveedores de servicios de telecomunicación se ven obligados a seguir, a duras penas, el ritmo de crecimiento y los cambios tecnológicos.

[28] http://www.ahciet.net/común/pags/agenda/eventos/2007/165/ponencias/JoseLuisPeralta.ppt

Por tanto, las redes IP se justifican, en primer lugar, por costos bajos en crecimiento y explotación y, en segundo lugar, por su flexibilidad a soportar nuevos y mejores servicios, que sigan posibilitando el mantenimiento de los ritmos de crecimiento que se experimenta en la actualidad.

1.4.2.1 Convergencia de redes unificadas

Las redes IP convergentes reducen los costos y la complejidad ya que una infraestructura común para las comunicaciones de voz y datos contribuye a reducir los costos de administración de sistemas, hardware y soporte. Al mismo tiempo, una red IP convergente ofrece una gama mucho más amplia de servicios con una mínima o nula inversión adicional.

La convergencia permite integrar la gestión de correo electrónico, mensajería instantánea, conferencias, presencia de usuarios y voz aumentando la productividad y creando nuevas prácticas laborales que aporten mayor valor.

Asimismo, una estructura unificada permite satisfacer con mayor eficacia las necesidades de acceso de los distintos usuarios de toda la empresa.

Este modelo de convergencia ofrece entre otras las siguientes ventajas:

- Menor capital, no se duplican las infraestructuras para voz y datos.
- Procedimientos de soporte y configuración simplificados.
- Mayor integración de las distintas ubicaciones.
- Fácilmente escalable.
- Unifica los medios modernos de comunicación en una sola aplicación.

1.4.2.2 Convergencia a redes IP

Las futuras redes convergerán en una sola y para ello usarán como base de sus arquitecturas y funcionamiento el protocolo IP.

Los beneficios son muchos, comenzando por que todo se manejaría por medio de ceros y unos, permite mayor precisión, rapidez y control de los servicios.

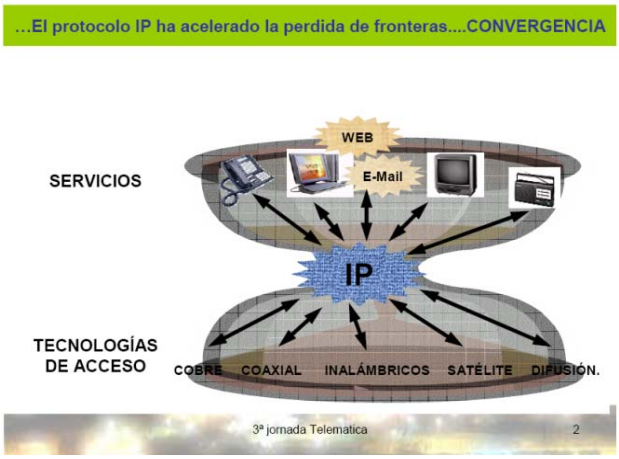


Fig. 1. 24. Importancia del Protocolo IP en la Convergencia [29]

Actualmente, este cambio está en plena evolución, prueba de ello es la mayor acogida y adquisición de teléfonos IP y centrales IP por parte de pequeñas, medianas y grandes empresas en todos los campos.

1.4.3 CONVERGENCIA Y MOVILIDAD

La convergencia de las redes celulares con las redes empresariales de datos abrirá las puertas a una gran gama de aplicaciones basadas en una movilidad total. El tener un canal de acceso a la red de nuestra empresa que sea móvil e independiente de nuestra ubicación pone a prueba el desarrollo de un gran número de soluciones que hasta el día de hoy están limitados por acceso a las redes.

La convergencia también se aprecia en los dispositivos. Es cada vez más clara la convergencia entre las agendas digitales y teléfonos celulares. Existen ya actualmente agendas digitales con cámara fotográfica, teléfonos con cámara fotográfica, o bien teléfonos con organizador electrónico o agendas digitales con

29

http://www.udistrital.edu.co/comunidad/eventos/jornadatelematica/articulos/3j/CONVERGENCIA%20ENTRE%20REDES%20FIJAS%20Y%20REDES%20MOVILES.pdf

teléfono celular. Sin embargo, muchos de estos dispositivos son limitados ya sea por una resolución pobre de las cámaras fotográficas, limitación en el espacio de almacenamiento, incompatibilidad de sistemas operativos, etc.

En conclusión, la gran convergencia que se está desarrollando no solo entre redes de voz, redes de datos y redes celulares sino también en los diferentes dispositivos, resulta un escenario que no podemos perder de vista. Esta gran multi convergencia traerá una gran variedad de soluciones y servicios que cambiarán los esquemas de trabajo que conocemos actualmente.

1.4.4 CONVERGENCIA PDH/SDH SOBRE ETHERNET

1.4.4.1 Antecedentes

En los últimos años, SDH ha logrado ganar un lugar en las redes metropolitanas y de acceso, gracias a su gran capacidad y a la alta fiabilidad de las estructuras en anillo. Hoy, de forma generalizada, se usan equipos SDH en las instalaciones de los clientes para ofrecer servicios de circuitos alquilados para empresas medianas y grandes.

La tecnología SDH es capaz de redireccionar el tráfico evitando los puntos de falla. Existe varios tipos de mecanismos de protección, esto depende mucho de la topología de la red, pero las ventajas de las rede SDH son muchas y se basan en su probada confiabilidad y capacidad de restauración.

Por otro lado, la tecnología Ethernet se ha impuesto como el estándar universal en las redes LAN. El hecho de que esta tecnología esté ya muy extendida ayuda a bajar los costos es gracias a la gran cantidad de dispositivos en el mercado, los precios de los puertos Ethernet en los switch o en los routers es muy reducido comparado con los enlaces de subida en modo paquete sobre SDH.

Ethernet es también ampliable, ya que el mismo puerto físico puede entregar, por ejemplo, desde unos pocos Mbit/s a 100 Mbit/s (Fast Ethernet) e incluso 1 Gb/s

(Gigabit Ethernet; GbE), sin embargo, el bajo costo y la alta flexibilidad no son los únicos factores importantes en la red de un proveedor, habitualmente las redes LAN no garantizan el ancho de banda requerido a un usuario particular.

Siempre se ha buscado la manera de cómo explotar las ventajas de Ethernet sin sacrificar totalmente la alta fiabilidad que garantiza SDH. Los nodos ópticos multiservicio combinan las ventajas de Ethernet con la alta y bien probada fiabilidad del SDH. Adaptando las tramas Ethernet en los contenedores virtuales SDH, es posible conectarse directamente a un conmutador mediante interfaces Ethernet.

Las tramas Ethernet generadas por el equipo de datos se montan en los recursos de transporte SDH mediante un adaptador de tasas. Esto permite a los operadores de telecomunicación adaptar la cantidad de ancho de banda en la red óptica para cada servicio, reduciendo así el costo por bit transportado. Entonces, se pueden proveer servicios de alta disponibilidad, garantizados por una protección de clase SDH, al mismo tiempo que se conserva la flexibilidad de Ethernet.

Actualmente, las tecnologías predominantes en las redes para entornos de área metropolitana son SDH y Ethernet. Por un lado, SDH está diseñada básicamente para dar servicios de voz y por lo tanto es ineficiente para transportar tráfico de datos, pero ofrece mecanismos de protección frente a fallos muy robustos. Por otro lado, Ethernet constituye una solución viable para el transporte de datos, pero no optimiza el ancho de banda disponible.

1.4.4.2 Convergencia de Ethernet sobre PDH

EoPDH (Ethernet over PDH) es una metodología normalizada para el transporte de las tramas originadas en Ethernet, sobre la infraestructura existente de telecomunicaciones de cobre para potenciar la tecnología de transporte establecida PDH. EoPDH es una de las diversas tecnologías de transporte

Ethernet que permite a los proveedores de servicios de telecomunicaciones ofrecer un servicio de transporte Ethernet.

EoPDH se basa en un conjunto de tecnologías y normas de telecomunicaciones que permiten a los carrier[30] hacer uso de sus amplias redes (PDH y SONET/SDH) y equipamiento para proporcionar nuevos servicios Ethernet. Además, EoPDH garantizan la interoperabilidad y la migración gradual de los servicios PDH a las redes Ethernet.

EoPDH transporta tramas Ethernet sobre enlaces T1, E1, o DS3. Las tecnologías utilizadas en EoPDH incluyen la encapsulación de tramas GFP, Mapeo Ethernet, concatenación virtual, esquema de ajuste de la capacidad del enlace, y Gestión de Mensajes (OAM). En los equipos que soportan EoPDH también se incluye el etiquetado de tráfico para la separación de la información en redes virtuales.

1.4.4.3 Convergencia de Ethernet sobre SDH

Ethernet sobre SDH (EoS) o Ethernet sobre SONET se refiere a un conjunto de protocolos que permiten el tráfico de Ethernet para ser transportados sobre redes de jerarquía digital sincrónica en forma eficiente y flexible.

Las tramas Ethernet que se enviarán en el enlace SDH se envían a través de un encapsulado en bloque (normalmente Generic Framing Procedure o GFP) este encapsulamiento ayuda a crear un flujo de datos sincrónicos de los paquetes Ethernet que son asíncronos. El flujo sincrónico de los datos encapsulados es luego pasado a través de un bloque de mapeo que usualmente utiliza concatenación virtual (VCAT) a la ruta de la secuencia de bits sobre una o más rutas de SDH. Como se trata de byte interleaved este proporciona un mejor nivel de seguridad en comparación con otros mecanismos de transporte Ethernet.

Después de atravesar las rutas SDH, el tráfico se procesa de forma inversa: la ruta de concatenación virtual procesa la información para recrear el flujo

[30] Carrier: Operador de servicios de telecomunicaciones proveedor de servicios de alto nivel

sincrónico de bytes original, seguido por desencapsulación para convertir el flujo de datos sincrónicos a un flujo asíncrono de las tramas Ethernet.

Procedimiento de Tramado Genérico (GFP)

Las técnicas de encapsulado tales como (GFP) se aplica para adaptar las ráfagas de tráfico asíncrono en corrientes de datos orientados a bytes, la cual es después mapeada en las conocidas tramas de SONET/SDH.

GFP agrega a las tramas de Ethernet una cabecera de GFP y un campo de chequeo de trama FCS antes de llenar los contenedores de SDH/SONET. El monitoreo de tráfico es un tema de aseguramiento, se busca que las tramas no se pierdan o se reciban fuera de orden.

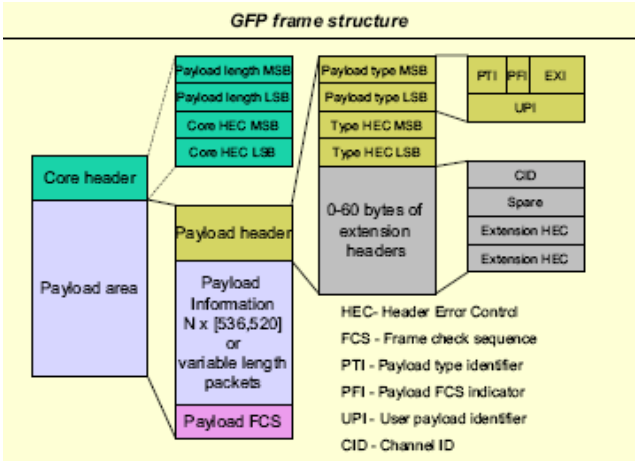


Fig. 1. 25. Estructura de Trama GFP [31]

Las cuatro partes que comprende la trama del procedimiento GFP son:

- Encabezado Principal, define la longitud de la trama GFP y detecta errores del CRC.
- Encabezado de Carga Útil, define el tipo de información transportada, ya sean tramas de administración o de clientes, así como el contenido del área de carga.
- Área de Carga Real, define la carga de transporte real.
- Campos opcionales de FCS para detección de errores.

[31] http://www.tkn.tu-berlin.de/curricula/ss02/vl-bbn/Folien/SS2002-IP_over_SONET-2up.pdf

Hay actualmente dos modos de señal del Cliente definidas para el procedimiento GFP:

- **GFP Enmarcado (GFP-F):** Donde cada trama de señal de datos del cliente es completamente mapeada a una única trama GFP. GFP-F se utiliza cuando la señal del cliente está empaquetada por el protocolo cliente.
- **GFP transparente (GFP-T):** Donde los códigos de los bloques 8B/10B de la señal de datos de varios clientes son los mapeados a una trama GFP periódica que maneja códigos de bloque 64B/65B.

Los servicios mapeados vía GFP-F utilizan la menor cantidad de encabezados para garantizar la mejor eficacia del ancho de banda, donde la prioridad de esos mapeos vía GFP-F es el transporte rápido y eficiente de los datos.

Adicionalmente al procedimiento GFP como mecanismo de adaptación, existen otros métodos. De estos, el protocolo de acceso de enlace (LAPS) y el control de enlace de datos de alto nivel (HDLC) son dos mecanismos de entramado predominantes.

2. CAPITULO II

ALGORITMOS DE COMPRESIÓN PARA REDES DE DATOS

2.1 GENERALIDADES

2.1.1 INTRODUCCIÓN

Actualmente la velocidad de procesamiento se incrementa rápidamente en comparación con la capacidad de almacenamiento y el ancho de banda de las redes de información. Para compensar esto, es común el procedimiento de reducir el tamaño de los datos, antes que incrementar la capacidad de almacenamiento y de transmisión de datos.

La compresión consiste en reducir el tamaño físico de bloques de información. Un compresor se vale de un algoritmo que optimiza los datos teniendo en cuenta el tipo de dato que se va a comprimir. Por lo tanto, es necesario un descompresor para reconstruir los datos originales por medio de un algoritmo opuesto al que se utiliza para la compresión.

El método de compresión depende exclusivamente del tipo de datos que se van a comprimir, no se comprime una imagen del mismo modo que un archivo de audio.

2.1.2 HISTORIA DE LA COMPRESIÓN

La historia de la compresión se remonta a los años 1960, en el que varios estudios buscaban codificar los símbolos de la manera más eficiente posible.

Inicialmente el planteamiento fue conseguir símbolos codificados en una tira de bits. Surge a partir de esto el *código ASCII* que utilizamos hoy en día.

Desde entonces, comienza la tarea para desarrollar un algoritmo que consiga el punto teórico de máxima eficiencia en la codificación de mensajes (*strings* o cadenas de caracteres).

David Huffman en el año 1954 realizó un trabajo que marcó la diferencia, este trabajo acabó siendo el conocido algoritmo sobre compresión de la Información, *Algoritmo de Codificación de Huffman*.

2.1.3 TIPOS DE COMPRESIÓN

2.1.3.1 La compresión física y lógica

La compresión física actúa directamente sobre los datos, entonces, es cuestión de almacenar los datos repetidos de un patrón de bits a otro.

La compresión lógica, por otro lado, se lleva a cabo por razonamiento lógico al sustituir esta información por información equivalente.

2.1.3.2 La compresión simétrica y asimétrica

La compresión simétrica, utiliza el mismo método para comprimir y para descomprimir los datos. De esta manera, cada operación requiere la misma cantidad de trabajo. En general, se utiliza este tipo de compresión en la transmisión de datos.

La compresión asimétrica demanda más trabajo para una de las dos operaciones. Es usual buscar algoritmos para los cuales la compresión es más lenta que la descompresión. Los algoritmos que realizan la compresión con más rapidez que la descompresión pueden ser necesarios cuando se trabaja con archivos de datos que se acceden con muy poca frecuencia, ya que esto crea archivos compactos.

2.1.3.3 La compresión reversible e irreversible

Se dice que una compresión es reversible si se sigue el procedimiento siguiente:

donde datos A = datos A

Para esto existen 3 técnicas principales:

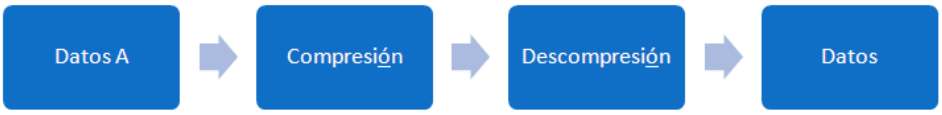
- Utilizar una notación diferente
  Similar a Abreviar
  Ejemplo: Para las ciudades del Ecuador uio, esm, gyq, cue
- Suprimir Secuencias Repetitivas
  Consiste en detectar secuencias de ciertos valores, ya sean caracteres o números.
  Ejemplo: Hooooooooola ---> H#10ola
- Asignar Códigos de Longitud Variable
  Se analiza la frecuencia de los valores, y se sustituyen los más populares por cadenas más cortas
  el ---> #3
  una ---> #25
  alguna ---> #456

Esta frecuencia se analiza a través del Código de Huffman, este determina las probabilidades de cada valor y construye un árbol binario que representa el código para cada valor, por ejemplo:

Letra	a	B	c	d	e	f	G
Frecuencia	0.4	0.1	0.1	0.1	0.1	0.1	0.1
Código	1	010	011	0000	0001	0010	0011

Tabla 2.1 Ejemplo de codificación por frecuencia

Se dice que una compresión es irreversible si se sigue el procedimiento siguiente:



De manera que el procedimiento de compresión irreversible es:

datos ---> compresión

Se emplea principalmente para ciertos tipos de archivos por ejemplo:

- Sonido: Modificando la codificación de modo que se alteren algunos parámetros, permitiendo cierto grado de distorsión.

- Imágenes: Al reducir la calidad o el tamaño.

Millones de colores

16 colores

- Video: Al reducir el número de frames por segundo.

Los datos multimedia (audio, video) pueden tolerar un cierto nivel de degradación sin que los órganos sensoriales (el ojo, el tímpano, etc.) distingan alguna degradación importante.

2.1.3.4 La codificación adaptiva, la semi adaptiva y la no adaptiva

Algunos algoritmos de compresión están basados en diccionarios para un tipo específico de datos, éstos son codificadores no adaptativos. La repetición de letras en un archivo de texto, por ejemplo, depende del idioma en el que ese texto esté escrito.

Un codificador adaptativo se adapta a los datos que va a comprimir. No parte de un diccionario ya preparado para un tipo de datos determinado.

Un codificador semi adaptativo crea un diccionario según los datos que va a comprimir, crea el diccionario mientras analiza el archivo y después lo comprime.

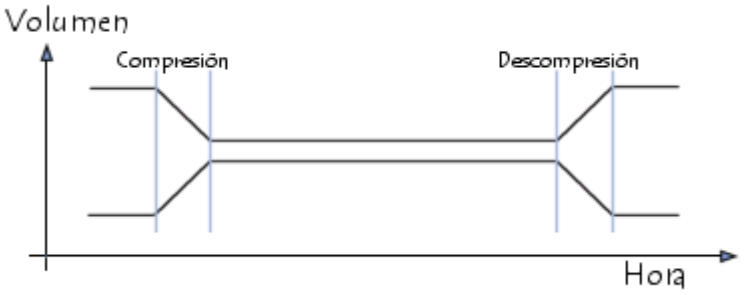


Fig. 2. 1. Compresión de datos [32]

2.2 ESTÁNDARES MÁS USADOS EN COMPRESIÓN EN EL DOMINIO IP

Antes de revisar los estándares de compresión sería relevante hablar de los tipos de datos a los cuales se los puede aplicar un proceso de compresión.

2.2.1 TIPOS DE DATOS

2.2.1.1 Datos multimedia

Los datos multimedia se han vuelto populares en la actualidad. Hoy se cuenta con sistemas que soportan texto, audio, imágenes y video. El poder de procesamiento de las computadoras permite manejar estas capacidades y además, los datos multimedia se comparten entre computadoras a través del Internet.

Literalmente, multimedia significa uno o más medios. Existen dos medios no continuos (textos e imágenes) y dos medios continuos (audio y video). Sin

[32] http://es.kioskea.net/video/compress.php3

embargo, si se habla de multimedia las personas entienden la combinación de dos o más medios continuos, es decir, que tenga que reproducirse por un determinado tiempo y de manera bien definida.

2.2.1.2 Texto

El texto es el medio más simple. Se compone por la secuencia de caracteres ordenados que tienen un significado. Existen estándares que representan los caracteres como números binarios, de esta forma, las computadoras entienden y representan estos caracteres. Uno de estos estándares es el código ASCII.

2.2.1.3 Imágenes

Las imágenes son cuadrículas de pixeles ordenados en un espacio delimitado. Cada pixel se representa con un color diferente y todos los pixeles en conjunto forman una imagen en la pantalla. Hay diferentes formas de representar un color, por ejemplo el estándar RGB.

Existen fotografías e ilustraciones de gran tamaño por lo que es necesario comprimirlos para transportarlos por Internet en un tiempo aceptable. Por ejemplo, un estándar que comprime fotografías con tonos continuos es JPEG.

2.2.1.4 Audio

La sensación del sonido es producida por una onda acústica. El oído humano puede percibir ondas acústicas dentro de un rango de frecuencias de 20 Hz a 20 KHz. El oído es muy sensitivo y detecta variaciones que duran unos milisegundos.

El ojo, en cambio, no percibe cambios de luz que duran lo mismo. Por estas razones, en transmisiones multimedia, la calidad en el sonido es más afectada que la calidad de las imágenes.

Un convertidor análogo digital, puede convertir las ondas de sonido en ondas digitales, tomando el voltaje eléctrico como entrada y generando un número

binario coma salida. De esta forma el sonido se convierte en datos que pueden ser transportados de un equipo a otro o a través de una red.

2.2.1.5 Video

El video es un medio que se percibe con la vista para obtener información por medio de imágenes. Cuando una imagen es proyectada sobre la retina, esta la retiene por unos milisegundos antes de perderse. Si se proyectan 50 o más imágenes por segundo, el ojo no detecta que está viendo imágenes discretas. Así se produce un efecto de movimiento.

Una cámara envía un rayo de electrones rápidamente a través de la imagen lentamente hacia abajo. Al final de la imagen, el rayo vuelve a empezar dibujando otra imagen. De esta forma se presenta la secuencia de imágenes en una pantalla.

El video se representa con una secuencia de frames. Cada uno se compone de una cuadrícula de pixeles. Un píxel puede ser un bit, para representar blanco o negro, o más bits para representar un rango más amplio de colores.

2.2.2 ESTÁNDARES DE COMPRESIÓN DE IMÁGENES

Se realizará una revisión de los diferentes estándares de compresión de imágenes que han sido establecidos por los diferentes organismos de estandarización.

Cada estándar utiliza una combinación diferente de tipos de compresión detalladas anteriormente.

2.2.2.1 Estándar de compresión para aplicaciones de fax

Este fue el primer estándar de compresión sin pérdidas, y fue desarrollado para aplicaciones de comunicaciones de fax. Los datos son escaneados en un formato blanco y negro y sus valores son representados por un bit por pixel.

Los factores de compresión alcanzados son de entre 20:1 a 50:1 para documentos y del orden de 1.5:1 para imágenes de escenas naturales.

2.2.2.2 Estándar JBIG

El estándar JBIG[33] de ISO, fue creado para mejorar los valores de compresión del estándar para aplicaciones de fax, logrando así una transmisión de imágenes de mayor calidad (niveles de grises).

JBIG es un estándar de compresión sin pérdidas que contiene un *modelador de probabilidades* y un *codificador aritmético*. El modelador es utilizado para estimar las probabilidades de aparición de cada muestra, que posteriormente utilizará el codificador aritmético para la asignación de palabras de código, según su frecuencia de aparición.

Este estándar alcanza entre un 20% y 50% de mayor compresión, para documentos, que el estándar para aplicaciones de fax. Para imágenes, el factor de compresión es entre 2 y 5 veces mayor.

2.2.2.3 Estándar JPEG[34]

La necesidad de un estándar de compresión para imágenes fijas tanto en blanco y negro como en color, fue estudiada hace unos cuantos años por un comité conocido como JPEG.

[33] JBIG.Joint Bi-level Image Experts Group, Estándar para compresión de imágenes
[34] JPEG. Joint Photographic Experts Group, Estándar para compresión de imágenes

JPEG tiene los siguientes modos de operación:

- **Codificación secuencial:** Cada imagen debe ser codificada en un único barrido de izquierda a derecha y de arriba a abajo.
- **Codificación progresiva:** La imagen se codifica en varios barridos para aplicaciones en las que el tiempo de transmisión es largo, y el observador prefiere ver como la imagen progresa de menos a más resolución en varias pasadas.
- **Codificación sin pérdidas:** La imagen se codifica garantizando que el algoritmo de codificación decodificará una imagen fiel a la original.
- **Codificación jerárquica:** La imagen se codifica en múltiples resoluciones, para que las capas con menor resolución puedan ser decodificadas sin tener que decodificar primero toda la imagen a resolución máxima.

La compresión JPEG puede efectuarse a diferentes niveles de compresión definidos por el usuario, lo que determina en qué medida una imagen deberá comprimirse. El nivel de compresión seleccionado está directamente relacionado con la calidad de la imagen solicitada.

2.2.2.4 Estándar JPEG2000

Otro estándar de compresión de imágenes fijas es JPEG2000. Fue desarrollado por el mismo grupo que desarrolló JPEG. El objetivo principal de uso son las aplicaciones médicas y la fotografía de imagen fija. Ofrece un rendimiento similar al JPEG pero con relaciones de compresión mucho mayores su rendimiento es ligeramente mejor que JPEG.

2.2.2.5 Estándar GIF [35]

GIF crea una tabla de 256 colores a partir de una de 16 millones. Si la imagen tiene menos de 256 colores, GIF puede almacenar la imagen sin pérdidas. Si la imagen tiene muchos colores, GIF usa algún algoritmo para aproximar los colores

[35] GIF. Graphics Interchange Format, Estándar para compresión de imágenes

de la imagen a una paleta limitada de 256 colores. GIF usa el color más cercano para representar cada píxel, y algunas veces usa un *error de difusión* para ajustar los colores de los píxeles vecinos y así corregir el error producido en cada píxel.

GIF produce compresión de dos formas. Primero, reduce el número de colores de la imagen a 256 y por tanto, reduce el número de bits necesario por píxel; después, remplaza áreas de color uniforme usando código de secuencias: en lugar de almacenar blanco, blanco, blanco almacena 3 blanco. Por lo tanto, GIF es una compresión de imágenes sin pérdida sólo para imágenes de 256 colores o menos.

2.2.3 ESTÁNDARES DE COMPRESIÓN DE VIDEO

2.2.3.1 Vídeo como una secuencia de imágenes M-JPEG [36]

Motion JPEG es el estándar utilizado más habitualmente en sistemas de vídeo IP. M-JPEG utiliza tecnología de codificación intracuadro, es decir, cada fotograma de una secuencia de video digital se comprime por separado como si fuese una imagen JPEG.

La cámara IP puede captar y comprimir, por ejemplo, 40 imágenes individuales por segundo (40 ips, imágenes por segundo), permite utilizar hasta 16 millones de colores (24 bits). Con una velocidad de imagen de aproximadamente 16 fps y superior, el visualizador percibe una imagen animada a pantalla completa. Esto implica aplicar el método de Motion JPEG o M-JPEG.

De la misma forma que cada imagen individual es una imagen JPEG completamente comprimida, todas ellas poseen la misma calidad garantizada, que se determina por el nivel de compresión elegido para el servidor de vídeo o cámara IP.

[36] MJPEG. Motion JPEG, JPEG Estándar para compresión de imágenes en movimiento,

2.2.3.2 MPEG [37]

MPEG es una de las técnicas de transmisión de audio y vídeo más conocidas, iniciado por el grupo Motion Picture Experts a finales de la década de los 80.

El MPEG utiliza códecs (codificadores-decodificadores) de compresión con bajas pérdidas de datos. En los códecs de transformación con bajas pérdidas, las muestras tomadas de imagen y sonido son fraccionadas en pequeños segmentos, transformadas y cuantificadas.

Los sistemas de codificación de imágenes en movimiento, tal como MPEG-1, MPEG-2 y MPEG-4, añaden un paso extra, donde el contenido de la imagen se predice antes de la codificación a partir de imágenes reconstruidas pasadas y se codifican solamente las diferencias con estas imágenes reconstruidas y algún extra necesario para llevar a cabo la predicción.

- **MPEG-1:** Salió al mercado en el año 1993 y su objetivo era el almacenamiento de vídeo digital en CD. Por tanto, la mayoría de códecs MPEG-1 están diseñados para una tasa de bits aproximado de 1,5Mbit/s. MPEG-1 permite un refresco de hasta 25 ips (PAL[38]) / 30 ips (NTSC[39]).
- **MPEG-2:** Se aprobó en 1994 como un estándar y fue diseñado para vídeo digital de alta calidad, TV digital de alta definición, soportes de almacenamiento de datos, vídeo de difusión digital y TV por cable. El proyecto MPEG-2 se centró en ampliar la técnica de compresión MPEG-1 con el fin de trabajar con imágenes más grandes y de mayor calidad a costa de una menor relación de compresión y una tasa de bits más elevada. La velocidad de imagen es de hasta 25(PAL) / 30 (NTSC) ips, como ocurre en MPEG-1.
- **MPEG-4:** Evoluciona de MPEG-2. Dispone de varias herramientas para reducir la tasa de bits necesaria para lograr cierta calidad de imagen.

[37] MPEG. Moving Picture Experts Group. Estándar para compresión de video
[38] PAL: Phase Alternating Line, Sistema de Codificación de señales de televisión analógica
[39] NTSC: National Television System Committee, Sistema de Codificación de señales de televisión analógica

Además, la velocidad de imagen no se limita a 25/30 ips. Hoy en día, la mayoría de las herramientas utilizadas para reducir la tasa de bits son relevantes para aquellas aplicaciones que no sean en tiempo real. De hecho, la mayoría de las herramientas en MPEG-4 que pueden utilizarse en una aplicación en tiempo real son las mismas herramientas que se encuentran disponibles en MPEG-1 y MPEG-2.

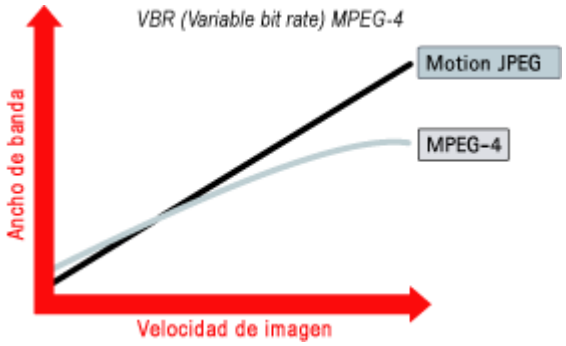


Fig. 2. 2. Comparación JPEG – MPEG-4 [40]

2.2.3.3 H.263

La técnica de compresión H.263 se centra en una transmisión de vídeo con una tasa de bits fija. La desventaja de tener una tasa de bits fija es que cuando un objeto se mueve, la calidad de la imagen disminuye. H.263 fue originalmente diseñado para aplicaciones de videoconferencia y no para aplicaciones de vigilancia donde los detalles son más importantes que una tasa de bits fija.

2.2.3.4 H.323

El estándar H.323 es utilizado para Voz sobre IP y para videoconferencia basada en IP. Es un conjunto de normas ITU para comunicaciones multimedia que hacen referencia a los terminales, equipos y servicios estableciendo una señalización en redes IP. No garantiza una calidad de servicio, y en el transporte de datos puede o no ser fiable.

[40] http://www.axis.com/products/video/compresion/index.htm

Estas redes son las que predominan hoy en todos los lugares, como redes de paquetes conmutadas TCP/IP, Fast Ethernet y Token Ring. Por esto, el estándar H.323 es un bloque de construcción para un amplio rango de aplicaciones basadas en redes de paquetes para la comunicación.

2.2.4 ESTANDARES DE COMPRESIÓN DE VOZ

2.2.4.1 G.711 Modulación por impulsos codificados (MIC) de frecuencias vocales

2.2.4.1.1 Introducción

G.711 es un estándar de la ITU-T para la compresión de audio. Este estándar es usado principalmente en telefonía. G.711 es un estándar para representar señales de audio con frecuencias de la voz humana, mediante muestras comprimidas de una señal de audio digital con una tasa de muestreo de 8000 muestras por segundo mediante tres operaciones fundamentales: Muestreo, Cuantificación y Codificación

Muestreo

Proceso en el cual se toman muestras de una señal analógica telefónica a intervalos de tiempo constantes. En primer lugar la señal telefónica que tiene una frecuencia vocal de 300 – 3400 Hz atraviesa un filtro, así se limita en ancho de banda de la señal de 0 a 4 Khz.

El siguiente paso es el de muestreo y retención, es decir se toma una muestra en un instante determinado, se necesita retenerla durante un cierto tiempo para poder medirla.

Según la teoría de Nyquist es necesaria una frecuencia de muestreo doble de la frecuencia máxima de la señal muestreada (2 x 3400=6800) se utiliza una frecuencia de muestreo de 8000 Hz.

Por tanto: T = 1/8000= 0,000125 seg = 125 μs

Fig. 2. 3. Muestreo de una señal analógica[41]

Cuantificación

Proceso mediante el cual a los valores muestreados (valores continuos) se le asignan valores discretos (valores digitales), dependiendo de la resolución del convertidor analógico-digital utilizado, en este caso de transmisión telefónica tiene una resolución de 8 bits. De este modo se consiguen 256 tramos de tensión o intervalos, llamados **intervalos de cuantificación.**

Estos intervalos se dividen en 128 para la parte positiva de la señal y 128 para la parte negativa. A todas las muestras cuya amplitud cae dentro del mismo intervalo se le asignan el mismo valor.

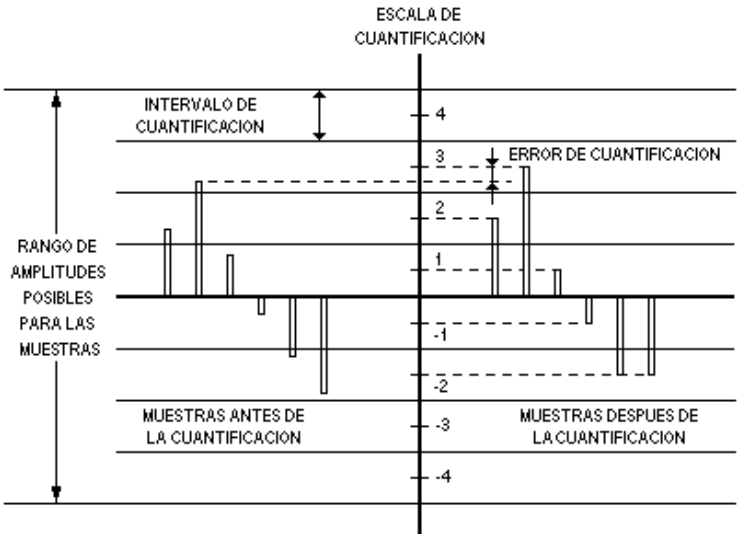


Fig. 2. 4. Cuantificación de una señal[42]

[41] http://www.terra.es/personal/ignaciorb/telefonia/mic/mic_1.htm
[42] http://www.terra.es/personal/ignaciorb/telefonia/mic/mic_1.htm

Cada intervalo de cuantificación está limitado por dos valores de decisión. Los valores de decisión límites de toda la gama de intervalos se llaman valores virtuales de decisión y determinan la máxima amplitud de señal que se puede transmitir sin recorte de crestas.

La UIT (Unión Internacional de Telecomunicaciones) recomienda para señales de audio vocales: la **ley de compresión A** en sistemas MIC europeos y **la ley de compresión μ** en sistemas MIC americanos.

LEY DE COMRESION LEY A

La ley A está formada por 16 segmentos de recta, de los cuales los cuatro centrales están alineados, por lo que se consideran uno sólo, reduciéndose los 16 segmentos a 13.

Cada uno de los 16 segmentos está dividido en 16 intervalos de cuantificación iguales entre sí, pero desiguales de unos segmentos a otros, excepto en los 4 segmentos centrales en los que son iguales todos los intervalos de cuantificación (tienen la misma pendiente).

En el eje de ordenadas, en los sistemas MIC europeos la gama de funcionamiento se encuentra dividida en 256 intervalos de cuantificación, 128 corresponden a muestras positivas y 128 corresponde a muestras negativas, que se agrupan, de 16 en 16, en 16 segmentos, 8 segmentos para muestras positivas y 8 segmentos para muestras negativas.

Normalmente a los cuatro segmentos de la parte central de la gama de funcionamiento se les considera un único segmento (el 7) de manera que la ley A se conoce como ley A de 13 segmentos.

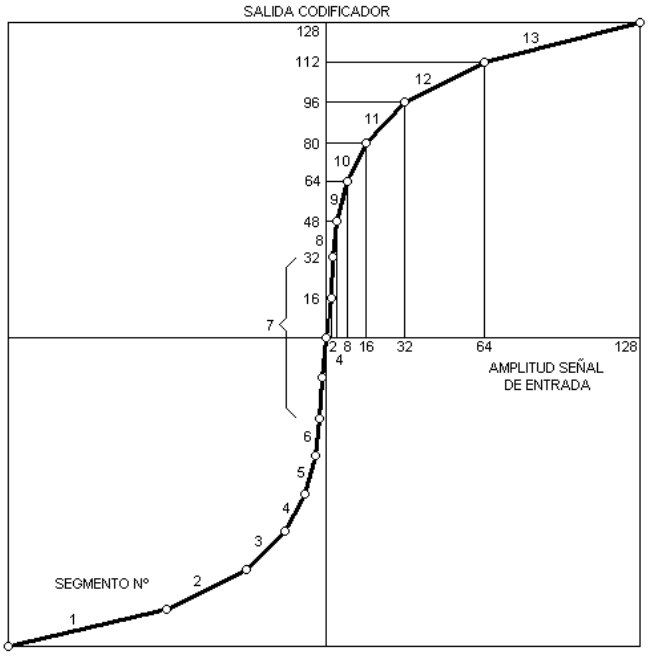


Fig. 2. 5. Ley A.[43]

Características de la Ley A

- Algoritmo estandarizado, definido en el estándar ITU-T G.711.
- Tiene una baja complejidad.
- Utilizado para aplicaciones de voz.
- Es usado en sistemas de transmisión TDM.
- No es recomendable para las transmisiones por paquetes.
- Factor de compresión aproximadamente de 2:1.

La Ley A asegura que los niveles altos de amplitud no necesitan tanta resolución como los bajos. Por lo tanto, si se asigna un mayor nivel de cuantificación a las bajas amplitudes y menor a las altas se obtendrá una mayor resolución, errores de cuantificación inferiores y por lo tanto una relación SNR (*relación señal-ruido*) superior que si se efectuara una cuantificación uniforme para todos los niveles de la señal.

[43] http://www.terra.es/personal/ignaciorb/telefonia/mic/mic_2.htm

El algoritmo Ley μ

Digitalmente, el algoritmo ley μ es un sistema de compresión con pérdidas en comparación con la codificación lineal normal. Esto significa que al recuperar la señal, ésta no será exactamente igual a la original.

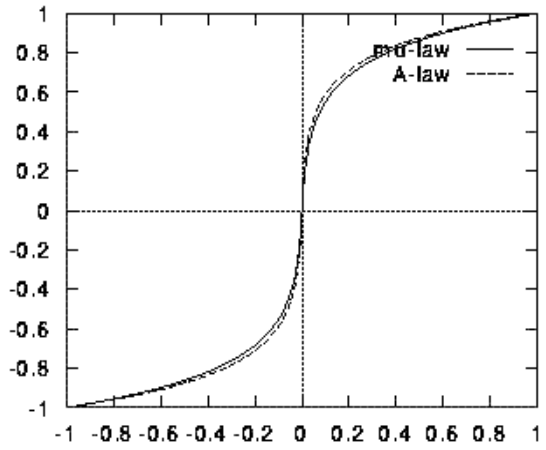


Fig. 2. 6. Ley de codificación Ley A y Ley μ [44]

Codificación

Con la codificación se representa una muestra cuantificada mediante un número binario.

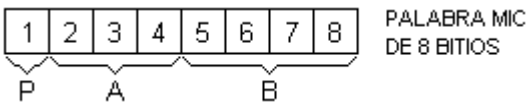


Fig. 2. 7. Palabra MIC codificada.[45]

El bit P indica la polaridad de la muestra positiva “1” negativa “0”

Los tres bits siguientes A, el código del segmento, 2^3=8, 8 segmentos positivos y 8 negativos.

[44] http://upload.wikimedia.org/wikipedia/commons/3/37/Mulaw.GIF
[45] http://www.terra.es/personal/ignaciorb/telefonia/mic/mic_2.htm

Los cuatros bits B el intervalo dentro de cada segmento, 2^4=16 intervalos en cada uno, desde el 0000 hasta el 1111.

2.2.4.2 Estándar G.723.1 códec de voz de doble velocidad para la transmisión en comunicaciones multimedios a 5,3 Y 6,3 kbit/s

2.2.4.2.1 Introducción

Este estándar es una extensión de la norma G.711 y G.721, posteriormente sustituida por la norma G.726, actualmente es obsoleta. Especifica una representación de códec que se puede utilizar para comprimir la voz u otras señales audio componentes de servicios multimedios a velocidad binaria muy baja.

Al diseñar este códec, la principal aplicación considerada fue la telefonía visual a velocidad binaria muy baja como parte de la familia general de normas H.324. G.723.1 se utiliza sobre todo en Voz sobre IP (VoIP), aplicaciones debido a su baja requisitos de ancho de banda.

Características

- Frecuencia de muestreo 8 kHz/16-bit (240 muestras de 30 ms de imagen).
- Tasa de bits fija (5,3 kbit/s y 6,3 kbit/s).
- Algorítmico retraso de 37,5 ms por imagen.

Velocidades binarias

Este códec tiene asociadas dos velocidades binarias. Se trata de 5,3 y 6,3 kbit/s. La velocidad más alta tiene mejor calidad, la velocidad más baja da una buena calidad y proporciona a los diseñadores de sistema más flexibilidad. Ambas velocidades son una parte obligatoria del codificador y del decodificador.

Descripción general

Este códec está diseñado para el funcionamiento con una señal de entrada analógica con el ancho de banda de telefonía, muestreándola a 8000 Hz y luego convirtiéndola a señal MIC lineal de 16 bits para su entrada en el codificador.

El codificador funciona con tramas de 240 muestras cada uno. Cada bloque pasa primero por un filtro y luego se divide en cuatro subtramas de 60 muestras cada una. Para cada subtrama, se calcula un filtro de códec de predicción lineal (PLC).

El filtro PLC para la última subtrama se cuantifica con un cuantificador vectorial de división predictiva (PSVQ, *predictive split vector quantizer*).

Los coeficientes PLC no cuantificados se utilizan para construir el filtro de ponderación de corto plazo, que se utiliza para filtrar toda la trama y obtener la señal de voz ponderada.

2.2.4.3 Estándar G.728 codificación de señales vocales a 16 kbit/s

2.2.4.3.1 Introducción

G.728 es un estándar de codificación que opera en 16 kbit/s. La tecnología utilizada es LD-CELP, (Low Delay Code Excited Linear Prediction)

Breve descripción del LD-CELP

Esta codificación opera con segmentos de voz de 0,625 mseg, correspondientes a 5 muestras PCM.

Por cada segmento de voz, el codificador analiza entre las 1024 vectores de su libro de códigos para encontrar la forma de onda del mismo que más se aproxime a la señal de entrada. Los 10 bits correspondientes al vector del libro de código

seleccionado son enviados al decodificador. De esta manera, cada 0,625 mseg el codificador envía 10 bits, lo que da una velocidad de 16 Kbps.

En la práctica, se usan 7 bits para representar 128 formas de onda patrón y los otros bits se utilizan para indicar la amplitud de la señal. Sabiendo que una señal analógica puede poseer una variedad infinita de valores la selección entre 1024 posibilidades se ve muy débil, y realmente lo sería si esta selección fuera estática.

Justamente la reputación de altamente compleja que posee la codificación CELP viene dada de la actualización constante de los libros de código y de los filtros, a partir del pasado reciente de la señal de entrada.

Al acumular solamente 5 muestras PCM para procesar el segmento de voz (en lugar de 80 para G.729) se logra tiempos de acumulación mucho menores que reducen el retardo del algoritmo, y, adicionalmente, resultan bloques de información más pequeños que se procesan de una manera mucho más rápida.

2.3 ESTRUCTURA

Inconvenientes que originan el uso de algoritmos de compresión:

- Insuficiente cantidad de espacio en medios de almacenamiento.
- Tiempos de transmisión prolongados y costos elevados.

Las computadoras por lo general codifican los caracteres mediante códigos como el ASCII y el EBCDIC. En el caso del ASCII, para la representación de un byte se cuenta con 8 bits, de manera que se tiene 2 a la 8 posibles representaciones de caracteres de longitud fija. Una posibilidad de disminuir el espacio, sería utilizando códigos de representación de longitud variable.

Así por ejemplo, para representar la letra a se utilizaría el valor binario 0, para la letra b con el valor binario 1, la letra c con el valor 01 y así sucesivamente. Pero para la codificación 0101 se puede interpretar como *abab* o como *cc*. El problema de decodificar un 01 lo solucionaría un prefijo que indique el comienzo y final de

un caracter antes de codificar el caracter, pero esto encarecería nuevamente el tema del espacio.

Vamos a analizar la estructura del Algoritmo de compresión de Huffman. Se elige el Algoritmo de Huffman puesto que es en base a este algoritmo que se va a desarrollar el software de simulación de compresión de datos (texto) en la presente investigación.

2.3.2 ALGORITMO DE HUFFMAN

El algoritmo de Huffman define un código variable para cada caracter sin la necesidad de utilizar un prefijo. De esta forma, los caracteres que son más frecuentes dentro de un alfabeto, se codifican con la menor cantidad de bits, mientras que aquellos que no sean tan frecuentes pueden tener una codificación más amplia.

Para ello, Huffman hace uso de una tabla de frecuencias donde se organizan los caracteres empleados en el texto y la frecuencia con que son utilizados.

Tomemos como ejemplo la frase: *esta es una prueba para tesis*

CARACTER	FRECUENCIA	CÓDIGO
-	5	011
A	5	111
E	4	001
S	4	101
T	2	110
U	2	010
P	2	0100
R	2	1100
N	1	0000
B	1	01000
I	1	11000

Tabla 2.2. Generación de códigos en base al algoritmo de codificación de Huffman

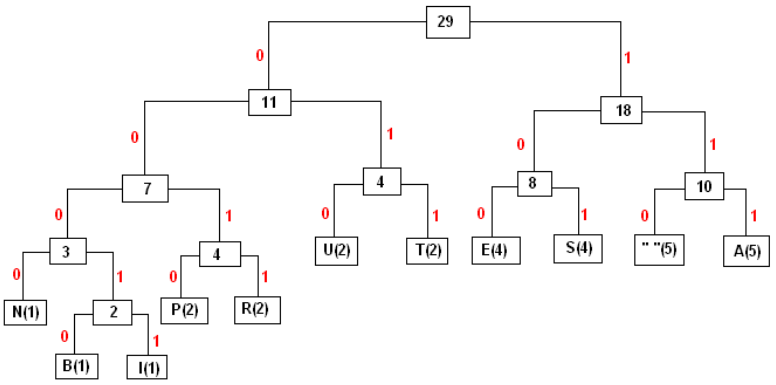
Con estos datos, ordenados de menor a mayor se obtiene la siguiente secuencia.
I(1) → B(1) → N(1) → R(2) → P(2) → U(2) → T(2) → S(4) → E(4) → A(5) → -(5)

A partir de aquí, se crea un árbol binario, construido de tal forma que siguiéndolo desde la raíz a cada una de sus hojas se obtiene el código Huffman.

Para ello se sigue los siguientes pasos:

- Se crean varios árboles, uno por cada uno de los símbolos del alfabeto, consistiendo cada uno de los árboles en un nodo sin hijos, y etiquetado cada uno con su símbolo asociado y su frecuencia de aparición.
- Se toman los dos árboles de menor frecuencia, y se unen creando un nuevo árbol. La etiqueta de la raíz será la suma de las frecuencias de las raíces de los dos árboles que se unen, y cada uno de estos árboles será un hijo del nuevo árbol. También se etiquetan las dos ramas del nuevo árbol: con un 0 la de la izquierda, y con un 1 la de la derecha.
- Se repite el paso 2 hasta que sólo quede un árbol.

Para nuestro caso se tiene el siguiente árbol.



Para obtener un código de un determinado caracter se debe ascender desde cada una de las hojas hacia la raíz. Si el ascenso es por la derecha se asigna un 1, si el ascenso es por la izquierda se asigna un 0. Para el caso de A(5), se tiene el código 111.

Una vez conocidos los códigos de cada caracter, se arma nuevamente la frase, sustituyendo el caracter por su respectivo código. Para el ejemplo propuesto, se tiene lo siguiente:

E	S	T	A	-	E	S	-	U	N	A	-	P	R	U	E	B	A	-
001	101	110	111	011	001	101	011	010	0000	111	011	0100	1100	010	001	01000	111	011

P	A	R	A	-	T	E	S	I	S
0100	111	1100	111	011	110	001	101	11000	101

Luego agrupando los bits en bytes, se tiene lo siguiente:

00110111	01110110	01101011	01101011	11011010	01100010	00101000	11101101	00111110	01110111	10001101	11000101

Este es el resultado de haber sometido la frase de nuestro al algoritmo de Huffman

Para decodificar se debe contar con la tabla de códigos que se genera luego de armar el árbol y con los bytes del mensaje codificado.

Tabla de Códigos

CARACTER	FRECUENCIA	CÓDIGO
-	5	011
A	5	111
E	4	001
S	4	101
T	2	110
U	2	010
P	2	0100
R	2	1100
N	1	0000
B	1	01000
I	1	11000

Tabla 2.3. Tabla de códigos en base al algoritmo de codificación de Huffman

Mensaje Codificado

00110111	01110110	01101011	01101011	11011010	01100010	00101000	11101101	00111110	01110111	10001101	11000101

Con esto se tiene:

E	S	T	A	-	E	S	-	U	N	A	-	P	R	U	E	B	A	-
001	101	110	111	011	001	101	011	010	0000	111	011	0100	1100	010	001	01000	111	011

P	A	R	A	-	T	E	S	I	S
0100	111	1100	111	011	110	001	101	11000	101

Y nuevamente se tiene el mensaje original.

2.4 VENTAJA Y DESVENTAJA DE LA COMPRESIÓN

VENTAJAS

- Almacenar sonido en forma digital, es almacenar datos, por tanto, pueden guardarse por años, sin que sufran deterioro alguno.
- El sonido digital posee una excelente relación señal ruido, es decir el ruido observado en grabaciones de casetes, aquí es descartado pues se elimina en el proceso de digitalización.
- Una señal digital es más fácil de transmitir, guardar o manipular, en el caso del sonido: editar, comprimir, etc.
- A la señal digital no le afecta el ruido. La señal digital no es tan sensible como la analógica a las interferencias que puede sufrir.
- La señal digital permite generar muchas copias sin pérdidas de calidad, en cambio con la señal analógica la copia es limitada y con pérdida de calidad.
- Ante la pérdida de cierta cantidad limitada de información, la señal digital puede ser reconstruida gracias sistemas de regeneración de señales. Las señales digitales, también cuentan con sistemas de detección y corrección de errores que, por ejemplo, permiten introducir el valor de una muestra dañada, obteniendo el valor medio de las muestras adyacentes (interpolación).

- La señal digital puede ser enviada a casi cualquier punto del planeta en cualquier momento a un muy bajo costo a través de Internet, esto sin que la señal sufra variaciones o alteraciones de calidad severas.

DESVENTAJAS

- Hay una pérdida forzada de información al convertir la información continua en discreta. Por mínimo e insignificante que sea, siempre hay un error de cuantificación que impide que la señal digital sea exactamente igual a la analógica que la originó.
- Si se utiliza compresión con pérdida, será imposible reconstruir la señal original idéntica, pero si se puede obtener una parecida, dependiendo de la calidad del muestreo tomado en la conversión de analógico a digital.

3. CAPITULO III

TECNOLOGÍA PDH

3.1 TÉCNICAS DE TRANSMISIÓN

La transmisión de datos puede clasificarse en asincrónica y sincrónica.

3.1.1 TRANSMISIÓN SÍNCRONA

En la transmisión de datos síncrona, existe una señal de reloj generada por un equipo centralizado y es la misma tanto para el emisor como para el receptor, permite que los datos se envíen a una tasa regular.

Algunas de las características de la transmisión síncrona son:

- Los bloques a ser transmitidos tienen un tamaño entre 128 y 1024 bytes.
- El rendimiento de la transmisión síncrona, cuando se transmiten bloques de 1024 bytes y se usan no más de 10 bytes de cabecera y terminación, supera el 99% (El rendimiento de transmisión es la relación que existe entre la información válida o información de datos y el volumen total de información que se transmite).

Ventajas y desventajas de la transmisión síncrona:

- Posee un alto rendimiento en la transmisión.
- Los equipamientos necesarios son de tecnología más completa y de costos más altos.

- Son aptos para ser usados en transmisiones de altas velocidades (iguales o mayores a 1200 baudios [46] de velocidad de modulación).
- El flujo de datos es más regular.

3.1.2 TRANSMISIÓN ASÍNCRONA

En la transmisión asíncrona, es el emisor el que decide cuando se envían los datos a través de la red. Por lo tanto, en una red asíncrona el receptor no sabe exactamente cuándo recibirá un mensaje.

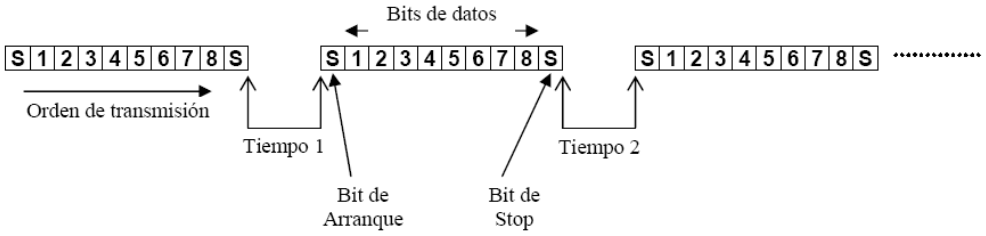


Fig. 3. 1. Modo de transmisión asíncrona [47]

Por lo consiguiente cada mensaje debe contener, aparte del mensaje, una información sobre cuando empieza y cuando termina el mensaje, de manera que el receptor conocerá lo que tiene que decodificar.

En el procedimiento asíncrono, cada caracter a ser transmitido es delimitado por un bit denominado bit de cabecera, y uno o dos bits denominados de bit de parada. El baud rate[48] (medido en baudios) define una tasa de reloj nominal, que es la máxima tasa de bit asincrónica.

Algunas de las características de la transmisión asíncrona son:

- Los equipos terminales que funcionan en modo asíncrono, se denominan también terminales en modo caracter.

[46] Baudios: Unidad informática que cuantifica el número de cambios de estado por segundo, que se producen durante la transferencia de datos
[47] http://www.eie.fceia.unr.edu.ar/~comunica/TBAApub/IntroTBAA.pdf
[48] Baud rate: Número de unidades de señal por segundo

- La transmisión asíncrona es usada en velocidades de hasta 1200 baudios.
- El rendimiento de usar un bit de arranque y dos de parada, en una señal que use código de 7 bits más uno de paridad (8 bits sobre 11 transmitidos) es del 72%.

Ventajas y desventajas del modo asíncrono:

- En caso de errores se pierde siempre una cantidad pequeña de caracteres, pues éstos se sincronizan y se transmiten de uno en uno.
- Bajo rendimiento de transmisión, dada la proporción de bits útiles y de bits de sincronismo, que hay que transmitir por cada caracter.
- Es un procedimiento que disminuye el costo del equipamiento pero la tecnología es menos sofisticada.
- Son aptos, cuando no se necesitan lograr altas velocidades.

3.2 ESTÁNDARES DE MULTIPLEXACIÓN PLESIÓCRONA Y SÍNCRONA

La problemática en el campo de las redes de comunicación es que los enlaces son caros, por esta razón hay que compartirlos entre varios usuarios, lo que disminuye notablemente el rendimiento de estos enlaces con la correspondiente reducción en las tasas de transferencia; la multiplexación pretende solucionar este inconveniente.

La multiplexación se define como una técnica que permite compartir un medio o un canal entre varias comunicaciones. Su objetivo es minimizar la cantidad de líneas físicas requeridas y maximizar el uso del ancho de banda de los medios.

Para aquello se utilizan los multiplexores y demultiplexores. Los multiplexores son equipos que reciben varias secuencias de datos de baja velocidad y las transforman en una única secuencia de datos de alta velocidad, que se transmiten hacia un lugar remoto.

En dicho lugar, otro multiplexor (que en este caso se denomina demultiplexor) realiza la operación inversa obteniendo de nuevo los flujos de datos de baja velocidad originales.

Fig. 3. 2. Componentes de un multiplexor [49]

MUX: combina los datos de n líneas de entrada y los envía por un único enlace de salida.
DEMUX: separa de 1 enlace a n salidas.

Básicamente existe tres estándares de multiplexación: espacio, frecuencia y tiempo. Esta enumeración coincide con el orden histórico en que fueron empleados en las redes de comunicaciones, y todos ellos se presentan en la capa física del modelo de referencia OSI.

3.2.1 MULTIPLEXACIÓN POR DIVISIÓN DE ESPACIO (SDM[50])

Un ejemplo de SDM es la transmisión por interfaz paralelo (ejemplo: centronic para impresoras) cuando se emplean múltiples conductores para interconectar dos equipos. En otras palabras, división de espacio significa físicamente separado.

En las redes telefónicas primitivas, un par de cobre conectaba a cada par de usuarios que mantenía una comunicación, aún de larga distancia, este es un ejemplo del primer uso de la división de espacio.

[49] http://www.tyr.unlu.edu.ar/cms/files/04-Multiplexaci%C3%B3n.pdf.
[50] SDM: Space Division Multiplexing: Multiplexación por división de tiempo.

En los sistemas de transmisión de datos primitivos, existía un cable separado para cada terminal hasta el computador central (host), esta forma de transmisión se volvió rápidamente impráctica.

3.2.2 MULTIPLEXACIÓN POR DIVISIÓN DE FRECUENCIA (FDM[51])

La multiplexación por división en frecuencia es una técnica que consiste en dividir el espectro de frecuencias del canal de transmisión mediante filtros y desplazar la señal a transmitir dentro del margen del espectro correspondiente mediante modulaciones, de tal forma que cada usuario tiene posesión exclusiva de su banda de frecuencias (llamadas subcanales).

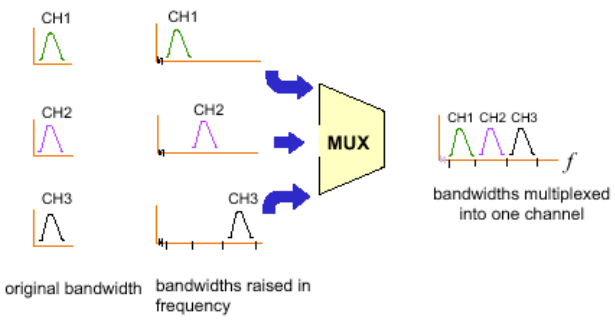


Fig. 3. 3. Multiplexación por división de frecuencia [52]

En el extremo de la línea, el multiplexor encargado de recibir los datos realiza la demodulación de la señal, obteniendo cada uno de los subcanales en forma separada. Esta operación se realiza de manera transparente a los usuarios de la línea.

Se emplea este tipo de multiplexación para usuarios telefónicos, radio, TV que requieren el uso contínuo del canal.

[51] FDM: Frecuency Division Multiplexing: Multiplexación por división de frecuencia.
[52] http://www.tyr.unlu.edu.ar/cms/files/04-Multiplexaci%C3%B3n.pdf.

Se pueden transmitir varias señales simultáneamente si cada una se modula con una portadora de frecuencia diferente, y las frecuencias de las portadoras están lo suficientemente separadas como para que no se produzcan interferencias.

Las interferencias que se pueden producir son: interferencia co-canal e interferencia por canal adyacente.

La interferencia co-canal se presenta en la misma banda de frecuencias que la señal útil.

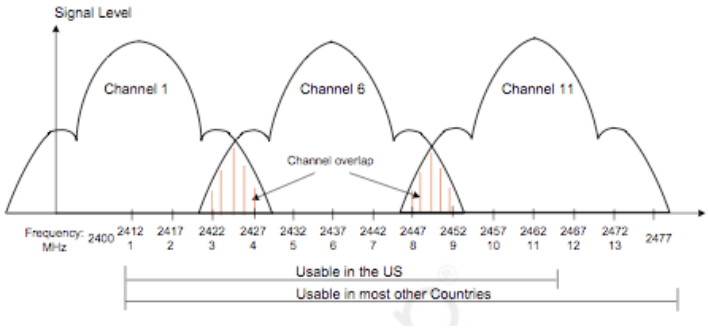


Fig. 3. 4. Interferencia Co-Canal [53]

La interferencia por canal adyacente se presenta por una señal en una banda distinta a la de la señal útil.

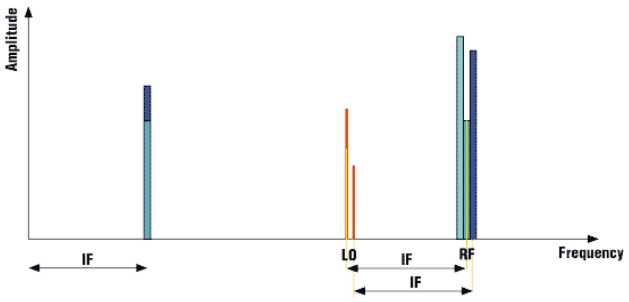


Fig. 3. 5. Interferencia por canal adyacente [54]

La señal que se transmite a través del medio es básicamente analógica, aunque las señales de entrada pueden ser analógicas o digitales. En el primer caso se

[53] http://www.analog.com/library/analogDialogue/archives/33-05/phase_locked/
[54] http://asoa.maldivesinfo.com/2007/11/14/interference-in-80211bg-networks/

utilizan las modulaciones AM[55], FM[56] y PM[57] para producir una señal analógica centrada en la frecuencia deseada. En las señales digitales se utilizan ASK[58] (modulación por desplazamiento de amplitud), FSK[59] (modulación por desplazamiento de frecuencia), PSK[60] (modulación por desplazamiento de fase) y DPSK[61] (modulación por desplazamiento diferencial de fase).

En el extremo receptor, la señal compuesta se pasa a través de filtros, cada uno centrado en una de las diferentes portadoras. De este modo la señal se divide otra vez y cada componente se demodula para recuperar la señal.

A medida que las redes de transmisión maduraron, se descubrió que era posible multiplexar muchas conversaciones analógicas en el mismo cable o banda de radio, modulando cada señal por una frecuencia portadora.

El espectro de frecuencia de la señal banda base fue separado en bandas de frecuencia. Este sistema marcó un aumento notable de eficiencia y funcionaba razonablemente bien para señales analógicas.

La tecnología estaba sin embargo limitada por la electrónica analógica y sufría de problemas de ruido, distorsión e interferencia entre canales (diafonía[62]) que complicaban las comunicaciones.

3.2.3 MULTIPLEXACIÓN POR DIVISIÓN DE TIEMPO (TDM)

La multiplexación por división de tiempo TDM[63], consiste en asignar a cada usuario la totalidad del ancho de banda disponible durante determinadas ranuras de tiempo. Esto se logra organizando el mensaje de salida en unidades de

[55] AM: Modulación por división de Amplitud
[56] FM: Modulación por división de Frecuencia
[57] PM: Modulación por división de Fase
[58] ASK: Modulación por desplazamiento de Amplitud
[59] FSK: Modulación por desplazamiento de Frecuencia
[60] PSK: Modulación por desplazamiento de Fase
[61] DPSK: Modulación por desplazamiento diferencial de Fase
[62] Diafonía: Perturbación, cuando parte de los datos presentes en una señal, aparece en el otro.
[63] TDM: Time Divison Multiplexing, multiplexación por division de tiempo

información llamadas tramas, y asignando intervalos de tiempo fijos dentro de la trama a cada canal de entrada. El primer canal de la trama corresponde a la primera comunicación, el segundo a la segunda, y así sucesivamente, hasta que el n-esimo más uno vuelva a corresponder a la primera.

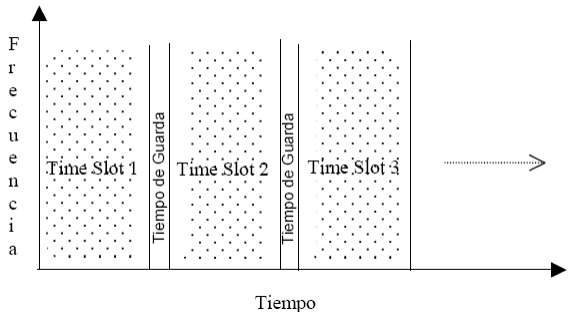


Fig. 3. 6. Multiplexación por división de tiempo [64]

El multiplexor por división en el tiempo muestrea, o explora, cíclicamente las señales de entrada (datos de entrada) de los diferentes usuarios, y transmite las tramas a través de una única línea de comunicación de alta velocidad.

Los TDM son dispositivos de señal discreta y no pueden aceptar datos analógicos directamente, sino demodulados mediante un módem.

Los sistemas de TDM convencionales emplean uno de los dos sistemas siguientes:

- Bit-Interleaved[65]
- Byte-Interleaved.[66]

3.2.3.1 Bit-Interleaved Multiplexing

Se reserva un intervalo de tiempo para cada salida al canal agregado. Cada intervalo, consta de un bit de cada uno de los canales de entrada, y siempre en el mismo orden, es decir, se intercalan los bits de cada uno de los canales de

[64] http://www.eie.fceia.unr.edu.ar/ftp/Comunicaciones/MUX.PDF.
[65] Bit Interleaved: Multiplexación por intervalo de bits .
[66] Byte Interleaved: Multiplexación por intervalo de byts.

entrada a la salida del mismo. Además hay un canal de sincronización, que transporta una señal fija que el receptor usa para sincronización.

El ancho de banda total es la suma de la de todos los canales de entrada menos el ancho de banda necesario para la sincronización. Este tipo de multiplexación necesita poco o nada de buffers. No se adapta a la transmisión de bytes.

3.2.3.2 Byte-Interleaved Multiplexing

En este tipo, lo que se intercala son octetos, y se envían de forma secuencial al canal agregado de alta velocidad. También se necesita un canal de sincronización para que los multiplexores funcionen de forma sincronizada.

Si los canales de entrada son todos síncronos, el ancho de banda total será la suma de todos los canales excepto el ancho de banda del canal de sincronización. Sin embargo si los canales son asíncronos, el ancho de banda agregado puede ser mayor si el tamaño del octeto agregado es menor que el tamaño del caracter asíncrono (bits de arranque + datos + bits de stop). La razón es porque los bits de arranque y de stop de cada octeto, son sustituidos antes de la transmisión, por lo que el receptor los debe restituir.

3.3 JERARQUÍAS DE MULTIPLEXACIÓN

En la transmisión de señales digitales se recurre a la multiplexación con el fin de agrupar varios canales en un mismo vínculo.

Si bien la velocidad básica usada en las redes digitales se encuentra estandarizada en 64 Kb/s, pues, la señal analógica se transforma en una secuencia binaria para ser transmitida por un canal digital, mediante técnicas de muestreo, cuantificación y codificación, las velocidades de los órdenes de multiplexación en cambio forman varias jerarquías.

Fig. 3. 7. Digitalización de un canal de voz analógico [67]

Existen tres jerarquías de multiplexación: europea, americana y japonesa, a las jerarquías mencionadas se las denomina Plesiócronas PDH porque el reloj usado en cada nivel de multiplexación es independiente de los otros niveles.

NIVEL JERÁRQUICO	VELOCIDAD DE TRANSMISIÓN (Kbit/s)/CANALES					
	EUROPA, SUDAMÉRICA		USA		JAPÓN	
0	64	1	64	1	64	1
1	2048	30	1544	24	1544	24
2	8448	30x4=120	6312	24x4=48	6312	24x4=48
3	34368	120x4=480	44736	48x7=336	32064	48x5=240
4	139264	480x4=1920	139264	336x3=1008	97728	240x3=720

Tabla 3.1. Jerarquías de multiplexación en PDH [68]

3.3.1 JERARQUÍA EUROPEA

La jerarquía europea, usada también en Latinoamérica, agrupa 30+2 canales de 64 Kb/s para obtener 2.048 Kb/s. Luego, por multiplexado de 4 tributarios sucesivamente, se obtiene las velocidades de 8.448 Kb/s; 34.368 Kb/s y 139.264 Kb/s.

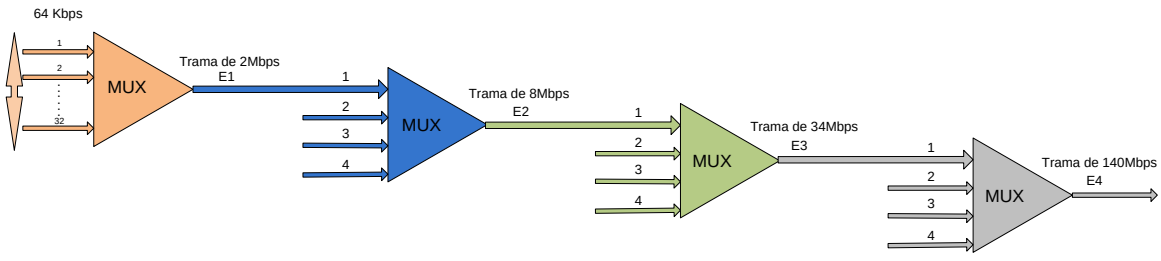


Fig. 3. 8. Jerarquía Europea de multiplexación [69]

[67] http://www.upseros.com/fotocopiadora/ficheros/Transmision%20de%20Datos/EST-2%20(curso%202003-2004)/Esquemas/06.%20multiplexacion%20mdf%20y%20mdt.pdf
[68] http://aniak.uni.edu.pe/sdemicro/Cap%2009%20MW%202005-1.pdf
[69] http://bieec.epn.edu.ec:8180/dspace/bitstream/123456789/531/7/T10455CAP1.pdf

3.3.2 JERARQUÍA NORTEAMERICANA

La jerarquía norteamericana agrupa en cambio 24 canales a una velocidad de 1.544 Kb/s. Posteriormente genera tres órdenes superiores (x4) a 6.312 Kb/s, (x7) a 44.736 Kb/s y (x6) a 274.176 Kb/s.

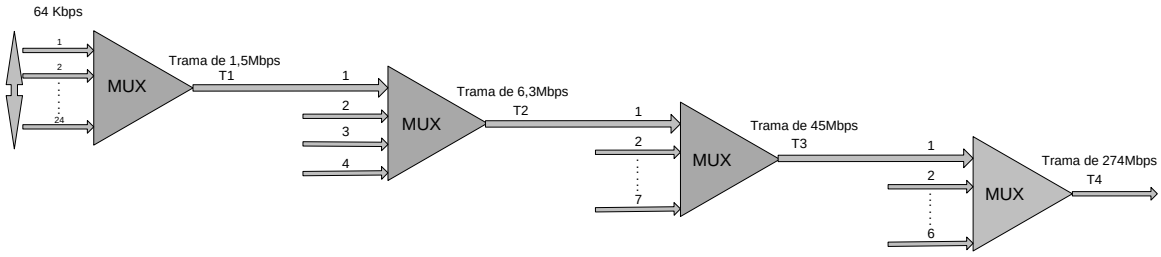


Fig. 3. 9. Jerarquía Norteamericana de multiplexación [70]

3.3.3 JERARQUÍA JAPONESA

La jerarquía japonesa recupera el valor de 6.312 Kb/s pero obtiene los órdenes jerárquicos de (x5) 32.064 Kb/s y (x3) 97.728 Kb/s.

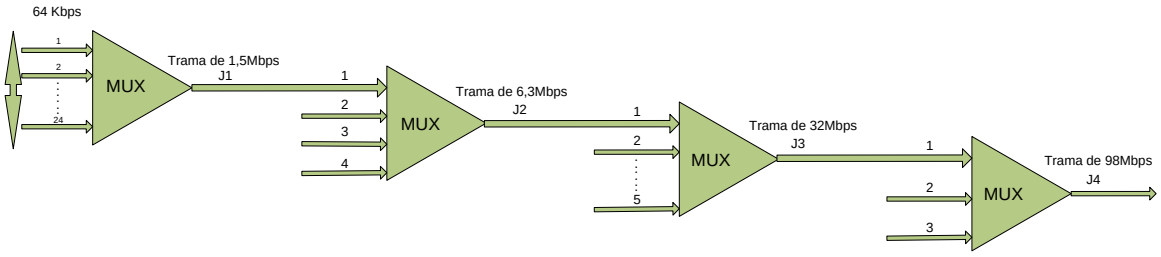


Fig. 3. 10. Jerarquía Japonesa de multiplexación [71]

3.4 NORMA EUROPEA ETSI

3.4.1 PRIMER ORDEN JERÁRQUICO

El primer orden jerárquico agrupa 30+2 canales de 64 Kbps, 30 canales telefónicos (1-15 y 17-31) y dos canales de control (0 control y 16 señalización).

[70] http://bieec.epn.edu.ec:8180/dspace/bitstream/123456789/531/7/T10455CAP1.pdf
[71] http://bieec.epn.edu.ec:8180/dspace/bitstream/123456789/531/7/T10455CAP1.pdf

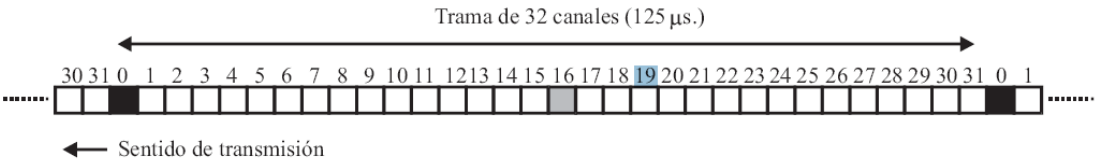


Fig. 3. 11. Primer orden jerárquico [72]

El primer orden jerárquico se multiplexa sucesivamente para obtener mayores velocidades y una multiplicación de la capacidad superior.

3.4.2 SEGUNDO ORDEN JERÁRQUICO

El segundo orden jerárquico agrupa cuatro sistemas de primer orden (120 canales telefónicos). La jerarquía plesiócrona correspondiente a 2048 Kb/s multiplexa en pasos de 4 entradas (tributarios de nivel inferior) para obtener la jerarquía superior.

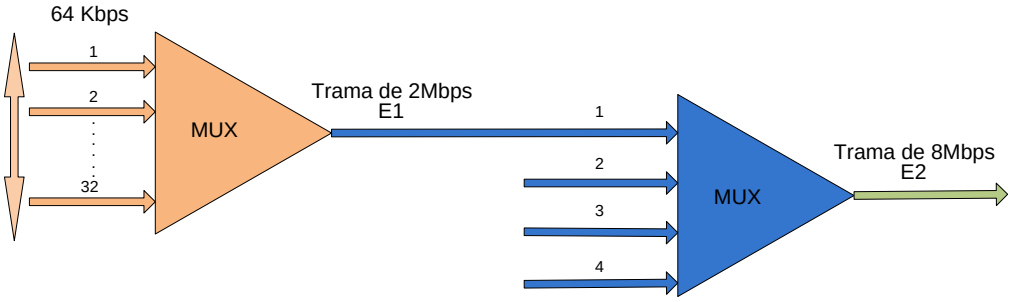


Fig. 3. 12. Segundo orden jerárquico [73]

3.4.3 TERCER ORDEN JERÁRQUICO

El tercer orden jerárquico agrupa cuatro sistemas de segundo orden (480 canales telefónicos). La jerarquía plesiócrona correspondiente a 8 Mbps multiplexa en pasos de 4 entradas (tributarios de nivel inferior) para obtener la jerarquía superior 34 Mbps.

[72] http://www.upv.es/bin2/caches/miw/visfit?=id419024&idioma=C
[73] http://bieec.epn.edu.ec:8180/dspace/bitstream/123456789/531/7/T10455CAP1.pdf

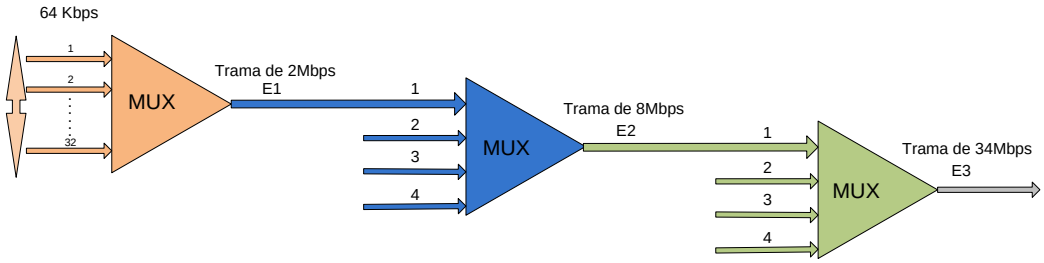


Fig. 3. 13. Tercer orden jerárquico [74]

3.4.4 CUARTO ORDEN JERÁRQUICO

El cuarto orden jerárquico agrupa cuatro sistemas de tercer orden (1920 canales telefónicos). La jerarquía plesiócrona correspondiente a 34 Mbps multiplexa en pasos de 4 entradas (tributarios de nivel inferior) para obtener la jerarquía superior 140 Mbps.

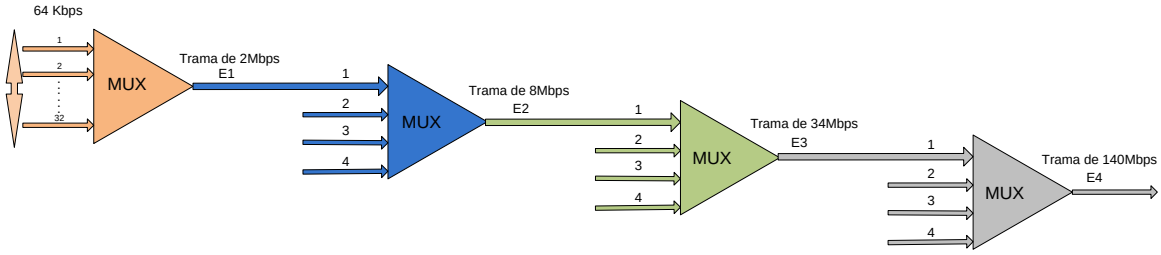


Fig. 3. 14. Cuarto orden jerárquico [75]

3.5 GESTIÓN DE REDES PDH

Se pueden diferenciar 3 generaciones de desarrollo en los sistemas de supervisión de redes digitales:

- Transmisión de alarmas: consiste en un sistema de multiplexación de alarmas sobre una trama de datos de baja velocidad (hasta 300 b/s). Se trata de una operación unidireccional desde las estaciones remotas hacia

[74] http://bieec.epn.edu.ec:8180/dspace/bitstream/123456789/531/7/T10455CAP1.pdf
[75] http://bieec.epn.edu.ec:8180/dspace/bitstream/123456789/531/7/T10455CAP1.pdf

un concentrador de alarmas. Estos sistemas actuaron hasta la década de los '80.

- Sistema de telesupervisión dedicado: permite el diálogo entre las estaciones remotas con un nodo. Permite efectuar la transmisión de alarmas, telecontroles, medidas a distancia y evaluación de la tasa de error BER. Estos sistemas comenzaron con la red digital PDH y se instalaron hasta mediados de la década de los '90.
- Red de gestión de telecomunicaciones: permite, además de las funciones indicadas, el almacenamiento de datos y la reconfiguración de la red, entre otras funciones. Lo interesante de esta red es que se encuentra normalizada por el ITUT para compatibilidad entre distintos productores y que permite la supervisión no solo de equipos de transmisión sino cualquier otro tipo de equipos. La velocidad de comunicación es sustancialmente más alta (nx64 Kb/s) debido al incremento de complejidad en el protocolo de comunicación.

La transmisión de alarmas (primera generación) representa una solución práctica pero insuficiente. La evaluación de la calidad del servicio (ITU-T G.821/826) es importante en los sistemas digitales. La gestión es una incorporación del último estado de evolución.

3.5.1. FUNCIONES DE LA TELESUPERVISIÓN PDH

Un sistema de telesupervisión consta de 3 grandes etapas:

- Estación Remota (Remote Station): Se ubican en las estaciones con equipamiento no atendidas o en aquellas con menor jerarquía. Las funciones de la estación remota incluyen la comunicación con la estación radioeléctrica para obtener alarmas, generar telecontroles, medir variables analógicas y medir la BER; por otro lado, la comunicación con la estación central para transferir dichas informaciones.
- Estación Central (Central Station): Se ubican en las estaciones terminales más importantes del sistema. Recibe los datos derivados desde las

estaciones remotas y realiza el procesamiento de los mismos para comunicarlos al operador del sistema de telesupervisión.

- Sistema de operación (Operating System): Consiste en una herramienta con capacidad gráfica de datos sobre la red.

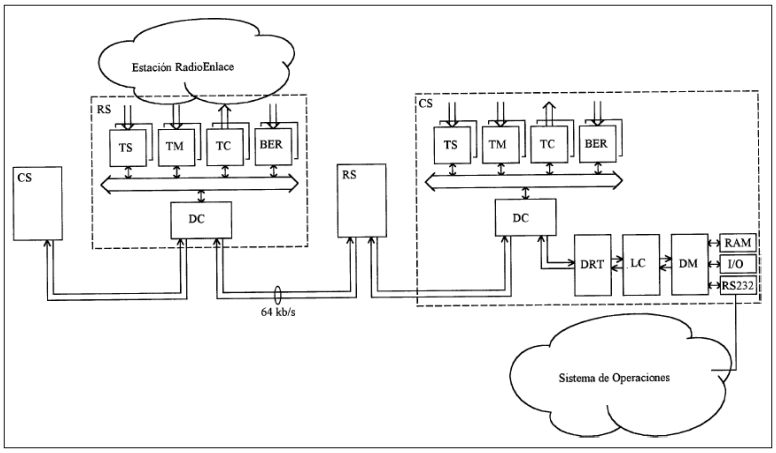


Fig. 3. 15. Estructura de una red de telesupervisión [76]

3.5.1.1 Estación remota

La estación remota consta de un microprocesador y tiene como función comunicarse con la red de supervisión y con la estación central. Una memoria EPROM guarda los programas a desarrollar y una memoria RAM retiene los datos del proceso.

El microprocesador, a través de un Buffer y un bus de datos, se comunica hacia 4 posibles unidades de entrada-salida con la red supervisada; las posibles interfaces son:

- **Unidad de Teleseñales:** Esta unidad permite leer el estado de las alarmas. Una alarma se considera activa cuando el valor de tensión se encuentra entre -2 y 0 V, siendo el valor de reposo -5 V (sin alarma). Se dispone de una memoria de forma que sólo se interpreten como alarma los cambios de estado cuya duración es superior a un determinado umbral (35 mseg para alarmas rápidas o 1 seg para alarmas normales). Sólo se cuenta un evento

[76] http://www.sincompromisos.com/Documentos/PDH-SDH/Gestion-PDH.pdf

por segundo. Es decir, cada segundo el micro procesador lee el estado de las alarma y lo interpreta; si existió más de un cambio en dicho segundo sólo se considera uno.

- **Unidad de Telemedidas:** Esta unidad convierte los valores analógicos de tensión (entre 0 y 5V) en un código binario de 8 bit que pueden ser interpretados en la estación central.
- **Unidad de Telecontroles:** Esta unidad permite gestionar datos y controles a distancia con un coste reducido a calidad de gestión, de esta manera se garantiza la calidad y eficiencia en la comunicación desde la estación central hacia cada una de las estaciones remotas.
- **Unidad de tasa de error (BER):** Esta unidad cuenta los errores que se obtienen desde alguno de los equipos de la estación. El conteo de errores se efectúa una vez por segundo y se evalúa. El microprocesador guarda la información obtenida de la comparación de la BER medida a fin de indicar los valores, una vez por hora la unidad del micro procesador transfiere desde la estación remota a la estación central los valores acumulados en la memoria RAM.

La memoria EPROM dispone de los programas que permiten:

- Evaluación periódica del estado de alarmas (se envían los cambios de estado a la estación central).
- Efectuar telecontroles y telemedidas (a pedido de la estación central).
- Desarrollar el Programa de comunicación entre la estación remota y la estación central (Pooling[77]).

El protocolo de comunicación entre las distintas estaciones centrales y las estaciones remotas de la red está de acuerdo con la arquitectura polling de la transmisión de datos. Consiste en un bus donde todas las estaciones transmiten sobre la misma línea, una a la vez, de acuerdo con la autorización que realiza la estación central. De las posibles estaciones centrales sólo una actúa de master-

[77] Pooling: Técnica de sondeo para el control en redes tipo LAN.

poller (genera el protocolo de comunicación Polling) mientras que las otras estaciones centrales se encuentran como listener a la espera de ser necesitadas.

3.5.1.2 Estación central

La base de la estación central la componen dos microprocesadores: uno de ellos se ocupa de la comunicación con todas las estaciones remotas y centrales y el otro se ocupa de la evaluación de los datos de la red y presentación al operador.

El microprocesador dispone de una memoria EPROM de programas y RAM para datos. En una estación central, este microprocesador máster desarrolla el programa de comunicación con las interrogaciones a cada una de las estaciones gestionadas. En una estación central listener el microprocesador se ocupa de actualizar los datos que circulan por la red de manera que cuando sea necesario pueda ingresar como máster con todos los datos idénticos a la estación central máster anterior.

El microprocesador evalúa sólo los cambios de alarmas de las estaciones supervisadas y los datos que ellas emiten cada hora.

La memoria EEPROM debe ser reprogramada con cada cambio de la estructura de la red. Esta memoria se carga mediante un programa de configuración que se encuentra en una computadora externa. La memoria RAM de datos puede que no sea suficiente para redes de ciertos tamaños y por lo tanto existe una unidad de expansión de memoria (128 ó 256 KBytes).

3.5.1.2 Software de telesupervisión

Una función importante del software del sistema es el protocolo de comunicación entre estaciones remotas y central. El modelo de conexión se reduce a 2 capas. En la capa física se define la interfaz física, en la capa enlace, se definen las tramas de comunicación y el control de errores.

La estación central máster envía la autorización de transmisión a cada una de las estaciones remotas mediante la trama de comando, al finalizar esta trama la estación identificada envía la trama de datos sobre la misma vía en la misma dirección. De esta forma la información pasa por todas las estaciones remotas y centrales lo cual permite actualizar los datos.

Cuando la red de datos se corta por alguna causa, otra estación central toma el control de aquella parte de la red asociada, convirtiéndose en estación central máster. La detección de la interrupción de la red se verifica si durante un cierto tiempo no se reciben datos en tránsito.

En la evolución de los sistemas de supervisión se observan las siguientes tendencias:

- Crecimiento en la velocidad de comunicación.
- Incremento en la complejidad del protocolo de comunicación.
- Tendencia a la normalización internacional.
- Tendencia a abarcar todos los sistemas de comunicaciones.
- Inteligencia distribuida de las unidades y equipos de red.

Las funciones específicas de la telesupervisión son:

- Transmitir el estado de las alarmas.
- Efectuar telecontroles.
- Realizar telemedidas.

3.6 VENTAJAS Y DESVENTAJAS

3.6.1 VENTAJAS

- PDH pretende solucionar la problemática de los altos costes que representan los enlaces en el campo de las redes.

- PDH es una técnica que permite compartir un medio o un canal entre varias comunicaciones.
- Minimiza la cantidad de líneas físicas requeridas y maximizar el uso del ancho de banda de los medios.
- La jerarquía PDH presente hoy en día tiene tres niveles de velocidad: serie Europea, serie Norte Americana y la serie Japonesa.

3.6.2 DESVENTAJAS

- El proceso de justificación no permite identificar una señal de orden inferior dentro de un flujo de orden superior sin demultiplexar completamente la señal de línea.
- Las tramas PDH disponen de poca capacidad adicional para el transporte de información de gestión.
- No permite mecanismos flexibles de reencaminamiento en caso de fallo.
- No existe un estándar mundial en el formato digital, existen tres estándares, el europeo, el estadounidense y el japonés.
- No existe un estándar mundial para las interfaces ópticas.
- Capacidad limitada de administración.

4. CAPITULO IV

TECNOLOGÍA SDH

4.1 ESTRUCTURA DE LA TRAMA SDH

4.1.1 INTRODUCCIÓN

En inicios de la década de los 80, el tráfico de voz era predominante sobre las redes de telecomunicaciones, con el paso de los años y el manejo de grandes flujos de información llega la necesidad de una comunicación de alta velocidad.

SDH es una tecnología dominante en la capa física de transporte de las redes actuales de fibra óptica. Tiene por misión transportar y gestionar gran cantidad de información proveniente de diferentes tipos de tráfico sobre la red, por lo tanto actúa como portador físico entre los niveles 2 a 4 (enlace de datos, red y transporte).

Se puede considerar a la transmisión SDH como canales que portan tráfico en forma de paquetes con información asociada para su correcta entrega en el destino, permitiendo el transporte de diferentes tipos de señales como voz, datos, video y multimedia.

4.1.2 CARACTERÍSTICAS SDH

SDH define elementos necesarios para llevar a cabo el transporte en la primera capa del modelo OSI. Los principales componentes que encontramos en cualquier sistema de transporte SDH son los siguientes:

- **Fibra óptica:** Es el medio físico común para las redes de transporte actuales por su mayor capacidad de soportar gran cantidad de información.

- **Multiplexación digital:** Permite que las señales de comunicación analógicas sean portadas en formato digital sobre la red, estas señales son muestreadas y cuantificadas convirtiéndose en sucesión de bits.
- **Topologías en anillo:** Estas topologías están siendo extendidas cada vez en mayor número, si un enlace se pierde hay un camino alternativo por el otro lado del anillo.
- **Gestión de la red:** Se ha desarrollado software que permite gestionar todos los nodos y caminos de tráfico desde un computador central.
- **Sincronización:** Los operadores de la red deben proporcionar temporización sincronizada a todos los elementos de la red para asegurar que la información que pasa de un nodo a otro no se pierda.

4.1.3 ESTRUCTURA BASICA DE SDH

SDH maneja una estructura básica llamada trama básica SDH, la cual tiene una duración de 125 microsegundos, y corresponde a una matriz de 9 filas y 270 columnas, cuyos elementos son octetos y como su duración es de 125 microsegundos su velocidad binaria será:

9 bytes x 270 bytes x 8000 muetras/s x 8 bits =155520 Kbps

Esta trama básica recibe el nombre de STM-1 (Synchronous Transport Module 1), en la trama se distinguen tres áreas: tara de sección (POH), punteros de AU y carga útil.

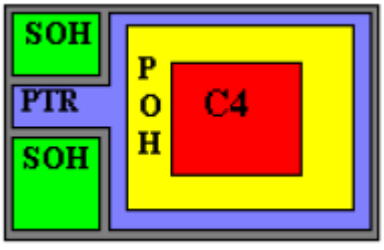


Fig. 4. 1. Estructura trama STM-1 [78]

[78] http://www.cec.uchile.cl/~jsandova/el64e/clases/sdh.pdf

Las velocidades de bit para los niveles más altos de las jerarquías SDH van de acuerdo al nivel N del Modulo de Transporte Síncrono (STM). Según la recomendación G.707 del CCITT estas velocidades son:

SONET		SDH	Tasa de bits
Nivel óptico	Nivel eléctrico	Equivalencia	(Mbps)
OC-1	STS-1	STM-0	51.84
OC-3	STS-3	STM-1	155.52
OC-12	STS-12	STM-4	622.08
OC-48	STS-48	STM-16	2488.32
OC-192	STS-192	STM-64	9953.28
OC-768	STS-768	STM-256	39812.12

Tabla 4.1 Comparación de tasas de SDH y SONET [79]

4.1.3.1 Terminología SDH

- **Contenedor (C):** Lleva la información, se define por niveles (n=1,2,3,4), dependiendo de la trama que sea, en el caso de una trama de 2 Mbps se almacena en un contenedor C-12. Tiene dos niveles que son 11 y 12, el segundo dígito indica si es un flujo de 1.5 Mbps (1) o 2 Mbps (2).
- **El Contenedor Virtual (Vc-n):** Es la estructura de transporte de la información a nivel de trayecto. Posee dos niveles que son: el VC-n de bajo nivel (n=1, 2) y el VC-n de alto nivel (n = 3, 4).
- **La Unidad Tributaria (TU-n):** Es la estructura que adapta la capa de bajo nivel y la de alto nivel, se forma por un contenedor virtual de orden 1, 2 ó 3 y un puntero que indica la posición del VC dentro de la entidad.
- **Grupo de Unidades Tributarios (TUG-n):** Constituido de varias unidades tributarias (TU), ocupando posiciones fijas y definidas en la carga de VC-n de alto nivel, tiene dos estructuras: TUG-2: formado por varios TU-1 o un TU2 y TUG-3: formado por varios TUG-2 o un TU3
- **La Unidad Administrativa AU-n (n = 3, 4):** Es la adaptación entre la capa de trayecto de alto nivel y la capa de línea, está formada por un VC de alto nivel y por un puntero que indica la posición del VC dentro de la entidad.

[79] http://www.eveliux.com/mx/redes-de-alta-velocidad-sdh-sonet.php

- **Grupo de Unidades Administrativas AUG:** Formado por varias unidades administrativas, ocupa posiciones fijas en el área de datos de una trama STM-N, puede ser formada por 3 AU-3 ó 1 AU-4.

4.2 ESTRUCTURA DE LA TRAMA STM-1, STM-n

4.2.1 TRAMA SDH STM-1

En SDH cada trama se encapsula en un contenedor, que incluye cabeceras de control para identificar el contenido de la estructura. Las tramas contienen la información de control cada nivel de la red, camino, línea y sección, además de la información de usuario (carga útil).

La información de usuario se agrupa para el transporte en los denominados Contenedores Virtuales, cada uno ofrece una determinada capacidad binaria, junto con la información de gestión de la misma Tara de Trayecto[80].

Los contenedores se multiplexan y agrupan con distintas jerarquías normalizadas, denominadas Unidades Tributarias y administrativas, para ser ubicados en la Carga útil de la trama STM-1, la que contiene una cabecera Tara de Sección[81] para la gestión del transporte y punteros hacia los contenedores.

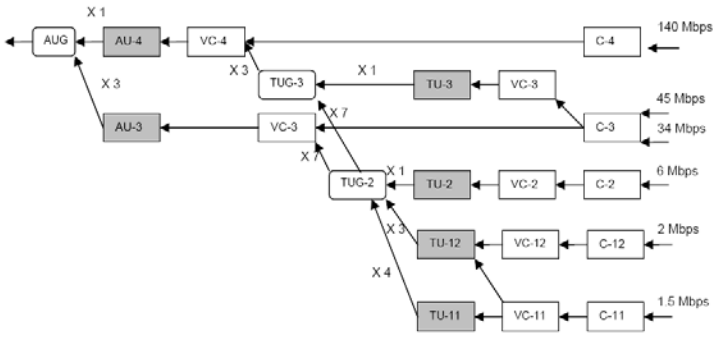


Fig. 4. 2. Mapa de estructura SDH [82]

[80] Tara de Trayecto: POH, monitorea la calidad e indica el tipo de contenedor SDH
[81] Tara de Sección: SOH, , soporta características de alineamiento de trama y mantenimiento y monitoreo de errores
[82] http://aniak.uni.edu.pe/sdemicro/Cap%2009%20MW%202005-1.pdf

4.2.1.1 Sección de cabecera SOH

El SOH es usado en el sistema de transporte individual para permitir el monitoreo de errores, la alarma de monitoreo y la administración de servicios y red. Contiene dos partes:

- Sección de regeneración de cabecera extra (RSOH)
- Sección de multiplexación de la cabecera extra (MSOH)

El RSOH está ubicado en cada regenerador, mientras que el MSOH en el multiplexor. Esto facilita el monitoreo del camino entre los multiplexores separadamente de las secciones de regeneración individual.

4.2.1.2 Cabecera de direcciones

Cada contenedor es ensamblado y desensamblado solo una vez. La cabecera de direcciones es transportado en el contenedor virtual entre los diferentes sistemas de transporte permitiendo el monitoreo del circuito de extremo a extremo.

Se definen dos tipos de cabecera de direcciones: cabecera de orden superior (niveles VC-3 y VC-4), y cabecera de orden inferior (niveles VC-2 y VC-12).

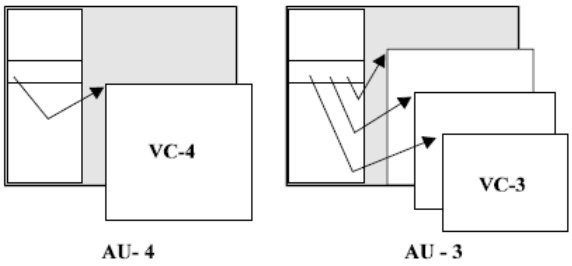


Fig. 4. 3. Posibilidad de carga VC4 [83]

[83] http://aniak.uni.edu.pe/sdemicro/Cap%2009%20MW%202005-1.pdf

4.2.1.3 Punteros

Un sistema sincrónico se basa en el hecho de que cada reloj está en sincronía de frecuencia con el siguiente. En una red, el reloj de frecuencia es extraído de la señal de línea, sin embargo puede existir en la señal variaciones de fase que deriven en una degradación de los datos transmitidos sobre la red.

Lo que realiza SDH para solucionar este problema es usar punteros para las direcciones del inicio en el contenedor de la trama.

- El puntero AU-4 muestra donde empieza el VC-4 en la trama.
- En los VC-4 están los punteros TU que muestran donde empiezan los VC de orden inferior (tales como VC-12), relativas a la posición de VC-4.

4.2.2 TRAMA SDH STM-N

STM-N es la estructura de información utilizada para transmitir información a nivel de sección, está formada por una cabecera (SOH) y por los datos del usuario, el campo de datos está formado por N grupos administrativos (AUG) situados en posiciones y definidas.

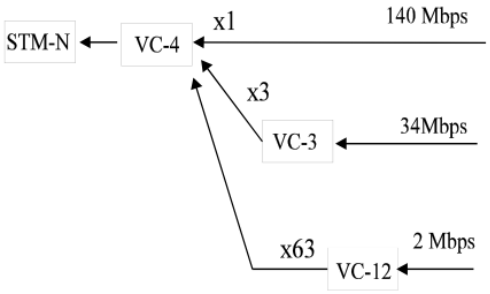


Fig. 4. 4. Trama STM-N [84]

[84] http://www.upseros.com/fotocopiadora/ficheros/Transmision%20de%20Datos/EST-22%20(curso%202003-2004)/Esquemas/06.%20multiplexacion%20mdf%20y%20mdt.pdf

4.3 ARQUITECTURA FUNCIONAL DE LA RED SDH

4.3.1 EQUIPAMIENTO SDH

Los equipos que intervienen en una red SDH consisten de cuatro bloques básicos:

- Regenerador.
- Un multiplexor terminal.
- Un multiplexor add-drop (ADM).
- Un switch de cruce de conexión (cross-conect).

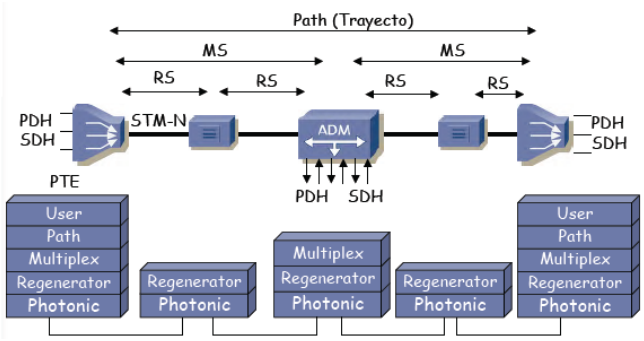


Fig. 4. 5. Elementos de una red SDH [85]

4.3.1.1 Regenerador

Se encarga de reconstruir una señal en base a relaciones de potencia y sincronismo, son elementos activos que derivan su señal de sincronismo de la trama de datos entrante. El espacio de red entre dos regeneradores se define como Sección de Regenerador y los equipos de esta sección se comunican a través de los canales existentes en RSOH encabezado de la sección de regenerador.

[85] https://www.tlm.unavarra.es/~daniel/docencia/rba/rba06_07/slides/16-TopologiasSDH.pdf

Fig. 4. 6. Bloque del regenerador [86]

4.3.1.2 Multiplexor terminal

Es el enlace SDH punto a punto, soporta ambos tráficos SDH o PDH. Combina varias señales de baja velocidad en una única señal de alta velocidad, con lo que se consigue una máxima utilización de de infraestructura física.

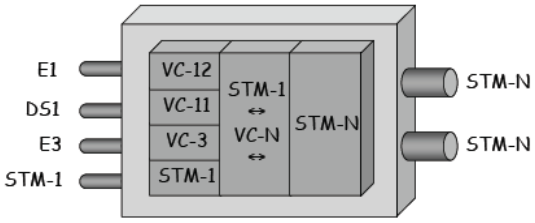


Fig. 4. 7. Bloque del multiplexor terminal [87]

4.3.1.3 Multiplexores sumadores

El ADM es un bloque fundamental de una red SDH. Se pueden crear interconexiones semipermanentes entre diferentes canales dentro de la red, esto permite que el tráfico sea enviado a nivel de contenedor virtual y si el operador de la red necesita cambiar los circuitos de tráfico y el encaminamiento puede conseguirse cambiando conexiones.

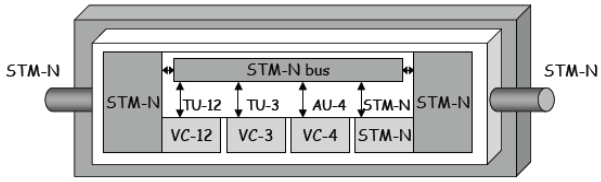


Fig. 4. 8. Bloque del multiplexor Add/Drop [88]

[86] https://www.tlm.unavarra.es/~daniel/docencia/rba/rba06_07/slides/16-TopologiasSDH.pdf
[87] https://www.tlm.unavarra.es/~daniel/docencia/rba/rba06_07/slides/16-TopologiasSDH.pdf
[88] https://www.tlm.unavarra.es/~daniel/docencia/rba/rba06_07/slides/16-TopologiasSDH.pdf

4.3.1.4 Conmutadores de cruce de conexión

Los conmutadores de cruce de conexión digital (DXC) son utilizados para tráficos de cruce de conexión, esto permite manejar el flujo en el tráfico SDH y potencia la capacidad de direccionamiento. El cruce de conexiones de alto grado permite la protección de los circuitos con fallas usando un sistema de protección de redes, este cruce de conexión se clasifica según la jerarquía de sus troncales de terminación (Trunk termination) y el nivel de cruce de conexión de los contribuyentes.

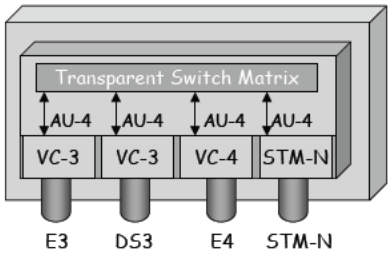


Fig. 4. 9. Bloque del multiplexor crossconector [89]

4.3.2 ARQUITECTURA DE RED SDH

La arquitectura es la organización de la topología de los elementos de la red y la interconexión entre los mismos. En SDH se asocia estos conceptos para definir los elementos y enlaces adecuados para la red.

4.3.2.1 Punto a Punto

Es el ideal para la conexión usuario-red, también red-red, marcando el comienzo y final de la trama. Para una red SDH se compone de dos multiplexores unidos por uno o dos enlaces STM-N, en cada uno de estos se arman y desmontan las tramas completas.

[89] https://www.tlm.unavarra.es/~daniel/docencia/rba/rba06_07/slides/16-TopologiasSDH.pdf

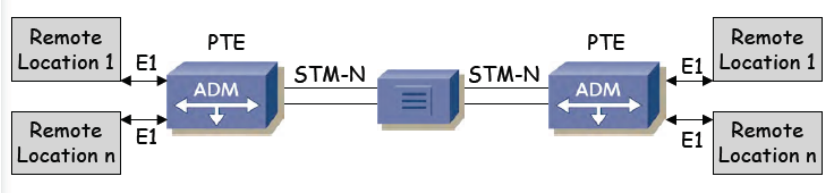


Fig. 4. 10. Topología Punto a Punto [90]

4.3.2.2 Lineales

Funcionan con una combinación de multiplexores add/drop, situados sucesivamente y terminan en cada extremo por un multiplexor terminal.

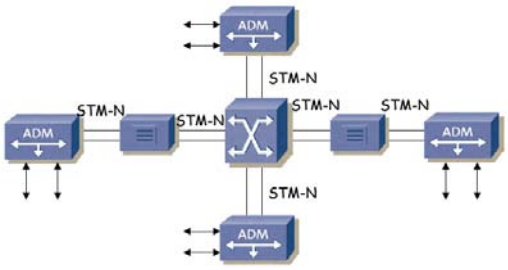


Fig. 4. 11. Aplicación lineal [91]

4.3.2.3 Anillos

Las redes SDH están formadas por varios anillos, con esto se toman sencillas decisiones de encaminamiento, están compuestas de un conjunto de multiplexores add/drop con dos enlaces STM-N unidos así se forma el anillo.

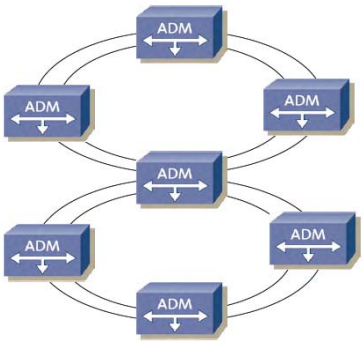


Fig. 4. 12. Aplicación en anillo [92]

[90] https://www.tlm.unavarra.es/~daniel/docencia/rba/rba06_07/slides/16-TopologiasSDH.pdf
[91] https://www.tlm.unavarra.es/~daniel/docencia/rba/rba06_07/slides/16-TopologiasSDH.pdf

Hay que insistir que se puede aprovechar los diferentes tipos de topologías, realizando diferentes combinaciones según los requerimientos de la red.

4.4 MULTIPLEXACIÓN SDH

Se entiende como multiplexar al hecho de utilizar la máxima capacidad o ancho de banda para transmitir múltiples señales simultáneamente por el mismo canal disponible.

Las tramas contienen información de cada uno de los componentes de la red, *trayecto*, *línea* y *sección*, además de la información de usuario. Los datos son encapsulados en contenedores específicos para cada tipo de señal tributaria.

A estos contenedores se les añade una información adicional denominada tara de trayecto POH (*Path overhead*), que consiste en una serie de bytes utilizados con fines de mantenimiento de red, y que dan lugar a la formación de los contenedores virtuales (VC).

El resultado de la multiplexación es una trama formada por 9 filas de 270 octetos cada una (270 columnas de 9 octetos). De las 270 columnas que forman la trama STM-1, las 9 primeras forman la denominada "tara" (*overhead*), independiente de la tara de trayecto de los contenedores virtuales antes mencionados, mientras que las 261 restantes constituyen la carga útil (Payload).

En la tara están contenidos bytes para alineamiento de trama, control de errores, canales de operación y mantenimiento de la red y los punteros, que indican el comienzo del primer octeto de cada contenedor virtual.

[92] https://www.tlm.unavarra.es/~daniel/docencia/rba/rba06_07/slides/16-TopologiasSDH.pdf

4.4.1 JERARQUIAS DE MULTIPLEXACIÓN SDH

La transmisión se realiza bit a bit en el sentido de izquierda a derecha y de arriba abajo. La trama se transmite a razón de 8000 veces por segundo (cada trama se transmite en 125 μs).

Por lo tanto, el régimen binario para cada uno de los niveles es:

Multiplexación STM-1: 8000*(270octeto s*9filas*8bits)=155 Mbps

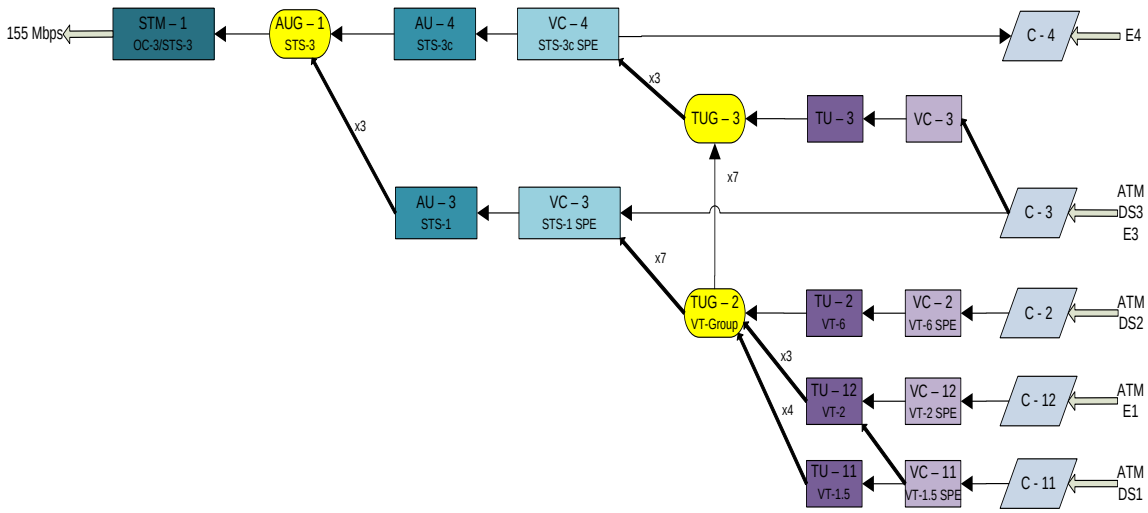


Fig. 4. 13. Multiplexación de Primer Orden [93]

[93] http://aniak.uni.edu.pe/sdemicro/Cap%2009%20MW%202005-1.pdf

Multiplexación STM-4: 4*8000*(270octetos*9filas*8bits)=622Mbps

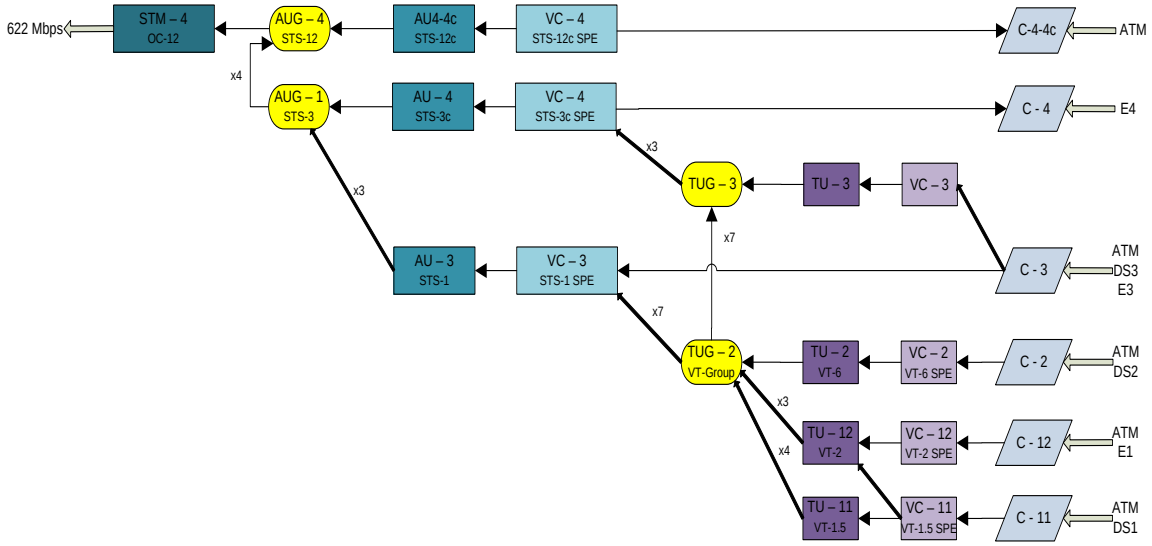


Fig. 4. 14. Multiplexación de Segundo Orden [94]

Multiplexación STM-16: 16*8000*(270octetos*9filas*8bits)= 2.5Gbps

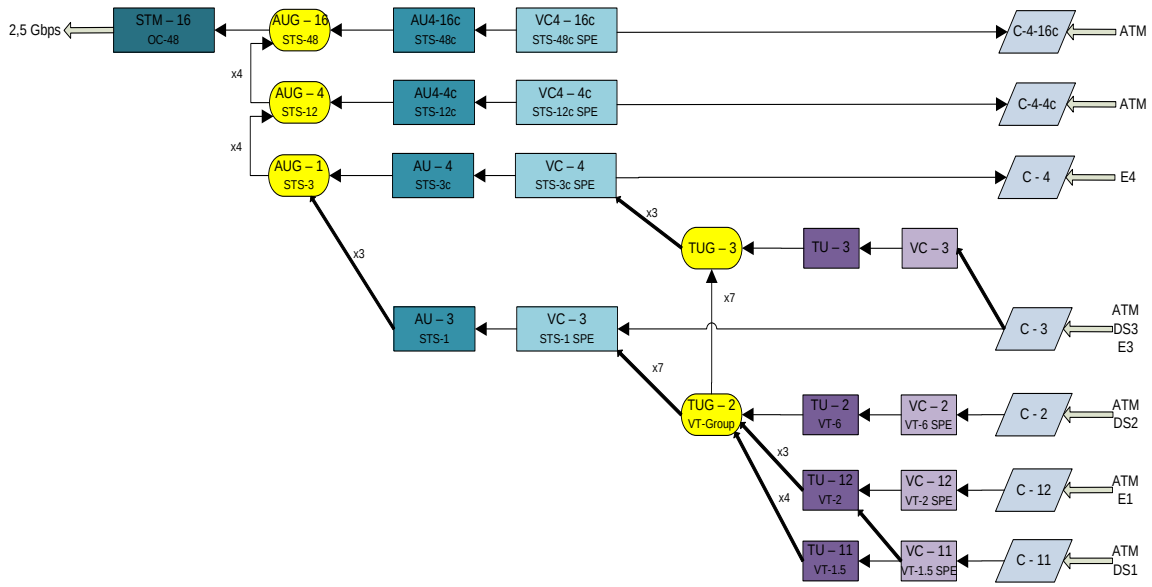


Fig. 4. 15. Multiplexación de Tercer Orden [95]

[94] http://aniak.uni.edu.pe/sdemicro/Cap%2009%20MW%202005-1.pdf
[95] http://aniak.uni.edu.pe/sdemicro/Cap%2009%20MW%202005-1.pdf

Multiplexación STM-64: 64*8000*(270octetos*9filas*8bits)= 10Gbps

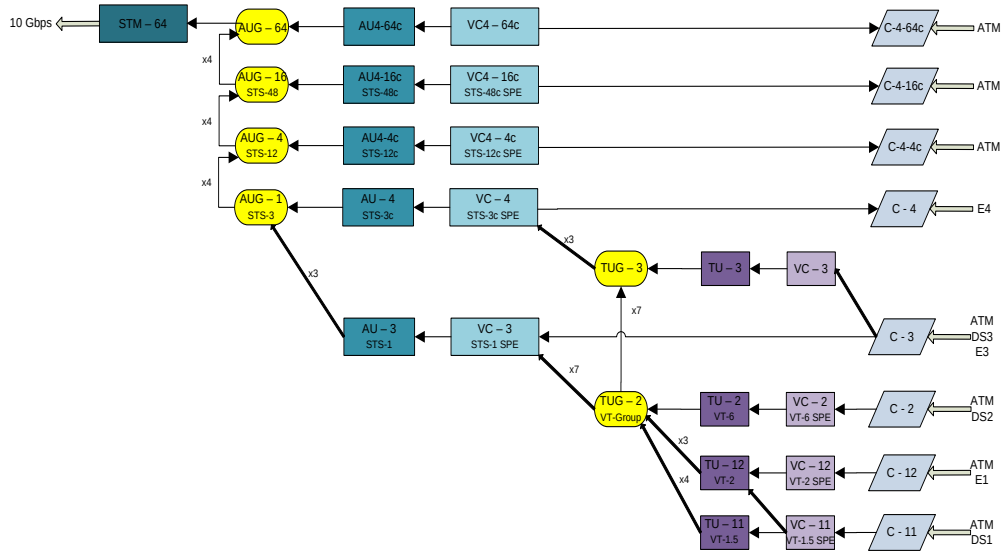


Fig. 4. 16. Multiplexación de Cuarto Orden [96]

Multiplexación STM-256: 256*8000*(270octetos*9filas*8bits)= 40Gbps

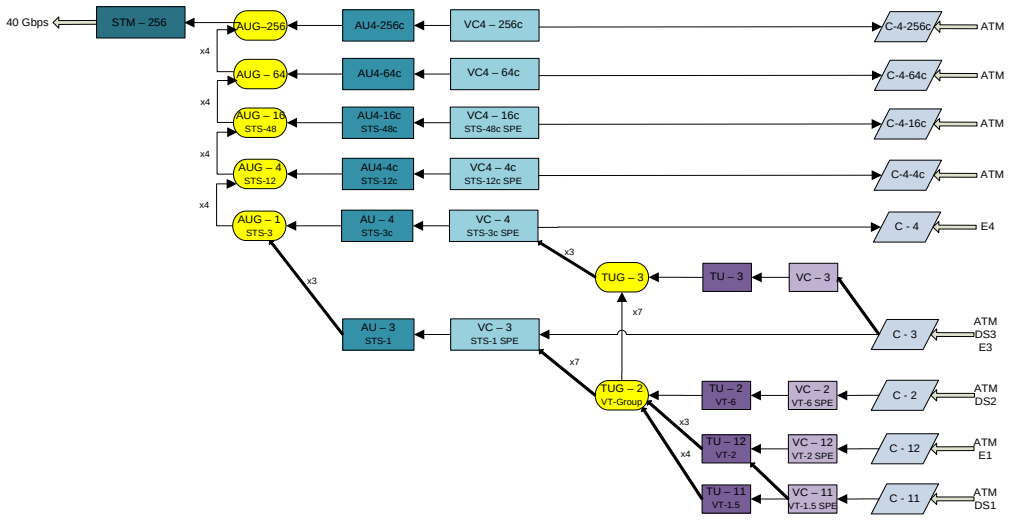


Fig. 4. 17. Multiplexación de Quinto Orden [97]

[96] http://aniak.uni.edu.pe/sdemicro/Cap%2009%20MW%202005-1.pdf
[97] http://aniak.uni.edu.pe/sdemicro/Cap%2009%20MW%202005-1.pdf

4.5 SINCRONISMO EN REDES SDH

Se entiende por sincronismo al proceso de hacer esclavo un reloj desde otra señal. En 1959 Bell desarrolló el proyecto Essex (*Experimental Solid State Exchange*) consistente en una central de conmutación digital con concentradores PCM y transmisión digital. Uno de los problemas descubiertos desde aquella época fue la sincronización de los centros de la red.

4.5.1 METODOS DE SINCRONIZACIÓN

En las redes digitales se mezclan las áreas internamente sincrónicas conectadas con áreas plesiócronas entre sí. Una clasificación de las formas de operación es la siguiente:

- Operación síncrona despótica: método subordinado, método jerárquico o externo.
- Operación síncrona mutua: con control uniterminal o control biterminal.

La sincronización despótica ocurre cuando un reloj asume el poder sobre los otros. En el método subordinado, conocido como maestro-esclavo, uno de los relojes actúa de maestro. En el método jerárquico existe un orden entre los relojes para ocupar la función de maestro en caso de falla. En el caso de reloj externo la sincronización se recibe desde afuera de la red.

La sincronización mutua permite eliminar el reloj maestro y hacer que cada uno de los relojes se sincronice con el valor promedio de todos los relojes entrantes al nodo. En el caso del control uniterminal se toma el valor medio entre los relojes entrantes y el local. El control biterminal en cambio, transmite la diferencia de fase medida en un nodo al otro, obteniéndose un control enlazado en ambos extremos.

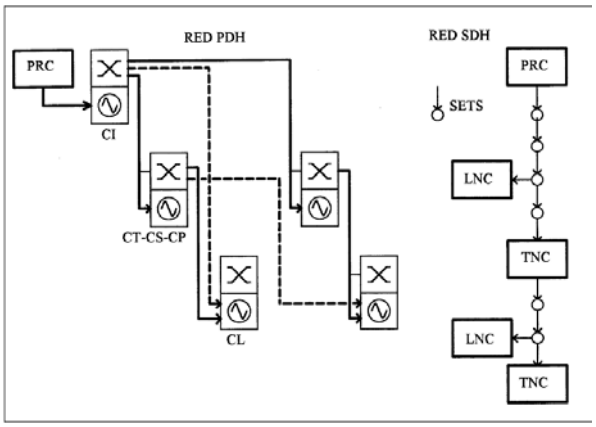


Fig. 4. 18. Mapa de sincronización [98]

Se observan que las redes pueden ser sincronizadas mediante una combinación compuesta por: centros de conmutación que funcionan con sincronización plesiócrona con relojes de alta estabilidad; centros nacionales regionales con sincronización despótica o plesiócrona jerarquizada y centros locales con sincronización despótica. Los relojes de estrato superior se sincronizarán mediante receptores GPS.

Las operadoras europeas adoptan la estructura jerárquica en tanto que en USA se prefiere la dependencia del GPS.

El funcionamiento de distintas redes se puede determinar de la siguiente forma:

- Modo Sincrónico, poseen el mismo reloj.
- Modo Pseudo-sincrónico. Consiste en 2 redes con reloj de estrato 1 en ambos casos.
- Modo Plesiócrono. Los elementos se encuentran en Holdover o Free running.
- Modo Asincrónico. El valor de offset de frecuencia es elevado (20 ppm para AIS).

98

http://www.rares.com.ar/albums/1_Manual_de_Telecomunicaciones_2000/408%20Relojes%20de%20temporizacion.pdf

4.5.2 SINCRONISMO INTRA ESTACIÓN

El Plan de distribución del sincronismo en una red parte desde los relojes primarios y mediante enlaces o anillos PDH/SDH se propaga hacia el exterior. Sin embargo, por razones de redundancia, se requieren relojes distribuidos y la reconfiguración en caso de corte. El plan de sincronismo toma en cuenta:

- El sincronismo dentro de una oficina de comunicaciones.
- Programación del sincronismo de los equipos de transmisión.
- Propagación de la temporización y reconfiguración de la red.

4.5.2.1 Sincronismo de equipos jerarquías PDH

Esta jerarquía tiene la diferencia que cada nivel de multiplexación posee un reloj independiente de los demás. Esta operación plesiócrona se repite entre transmisión y recepción. Los tributarios de entrada a cada nivel de multiplexación son embebidos dentro de la trama mediante el mecanismo de justificación positiva

4.5.2.2 Sincronismo de equipos jerarquías SDH

En PDH lo normal es el sincronismo interno, los equipos SDH se configuran para recibir la temporización desde el exterior en condiciones normales de funcionamiento. Solo en caso de falla se pasa al sincronismo interno. Una particularidad de gran importancia en SDH es que todos los equipos multiplexores y de transmisión se encuentran sincronizados desde la misma fuente.

4.5.2.3 Programación del sincronismo

Cada equipo puede tener varias alternativas de sincronismo. Las características de la gestión del sincronismo son:

- Programación por separado del reloj de sistema y el reloj de salida exterior.
- Programación por software de distintas fuentes de sincronismo.

- Se programan alternativas y prioridades de sincronismo para el reloj del sistema.
- Selección automática de la fuente de sincronismo en caso de falla.
- Configuración del Byte S1 para la reconfiguración del sincronismo en caso de falla.

4.5.3 SINCRONISMO INTER-ESTACIÓN

En los equipos PDH el proceso de justificación produce un error de fase de 1 bit por trama. El desplazamiento de punteros en SDH produce en cambio una variación de fase de 3 Bytes simultáneos.

El corrimiento en SDH se torna intolerable si existen cambios de punteros frecuentes en el mismo sentido. Algunos equipos dimensionan las memorias para soportar 2 corrimientos de punteros sucesivos en el mismo sentido. Por esta razón, se hace necesario sincronizar todos los equipos SDH para reducir el movimiento de punteros.

4.5.3.1 Propagación del sincronismo en PDH

La propagación del sincronismo en tramos de enlace PDH se realiza de la siguiente forma:

- Se toman como referencia los tributarios de 2 Mbps embebidos en la trama de orden jerárquico superior.
- Cuando la señal que transporta la sincronización es una trama de 2 Mbps en los tramos PDH, puede ser conveniente no cargarla con tráfico. Se evita que por razones de operación se modifique el enrutamiento por descuido en un cross-connect o add-drop intermedio.

4.5.3.2 Propagación del sincronismo en SDH

Para evitar el Jitter introducido por el salto de punteros la propagación de la temporización cumple las siguientes reglas:

- En los tramos de red SDH la temporización se extrae desde la trama STM-N de línea óptica (o desde STM-1 en un radioenlace). Se debe tomar desde la señal de línea en lugar desde un tributario de 2 Mbps incluido en la trama.
- Los equipos cuentan de una salida de reloj al exterior el cual se utiliza para llegar hasta la central de conmutación.

4.5.3.3 Loop de sincronismo

Al desarrollar el Plan de Sincronismo hay que evitar los posibles loop de temporización cerrado. Se destacan las siguientes particularidades:

- Los loop de sincronismo produce regiones aisladas de temporización con inestabilidades causadas por la realimentación.
- Para evitar el loop de sincronismo pueden usarse los mensajes de gestión (Byte S1) y relojes TNC con Hold-over.
- En una estructura en anillo deben existir al menos 2 nodos con relojes TNC. Las vías de acceso a ellos deben ser distintas.
- La subordinación a GPS permite disponer de varias referencias primarias distribuidas.

El loop se produce cuando debido a la pérdida de la referencia primaria un equipo Add-Drop conmuta desde el sincronismo externo al estado de sincronismo interno. La siguiente estación continúa sincronizada con la señal de recepción y se forma un loop. Para evitarlo se utiliza el mensaje de sincronismo SSM (Byte S1).

4.6 GESTIÓN DE REDES SDH

La tercera generación de sistemas de supervisión permite efectuar las operaciones de la segunda más otras adicionales (por ejemplo, reconfiguración dentro de una red en anillo). Posee una velocidad de comunicación y una capacidad de memoria mayor. A continuación en la Tabla 4.2 se comparan ambos sistemas.

Modelo del Sistema de Gestión	PDH	SDH
Funciones	Telesupervisión	Red TMN
Alarmas, Control, G.821	Si	Si
Configuración de red	No	Si
Protocolo de comunicación	Polling	HDLC
Velocidad de comunicación	64 Kbps	192 y 576 Kbps
Canal de comunicación	Independiente	SOH en STM-1
Unidad de supervisión	Separada	Integrada
Periféricos previstos	RS-232	LAN Ethernet
Interfaz y software	Propietario	Normalizados

Tabla 4.2 Comparación entre sistemas de gestión PDH y SDH [99]

La arquitectura típica del sistema de gestión para redes sincrónicas contiene los siguientes componentes:

- **Elementos de Red (NE):** Se considera como elementos de una red SDH a los multiplexores terminales o Add-Drop, los equipos terminales de línea o repetidores, los circuitos Cross-Connect, los equipos de radioenlace y las fuentes de sincronismo.
- **Adaptadores de interfaz:** Los elementos de red poseen hacia el exterior las interfaces F y Q que permiten la conexión con el sistema de operaciones. La interfaz F admite la conexión de una PC como sistema de gestión local. La interfaz Q, permite adaptar un elemento de la red NE ya existente a la TMN. La interfaz Q3 es normalizada y la Qx es propietaria del fabricante.

[99] http://www.scribd.com/doc/6538516/07-Gestion-de-Redes-Sdh

- **Elemento de Mediación:** Permite la conexión entre el elemento de red y el sistema de operaciones mediante un canal de comunicación de datos normalizado.
- **Sistema de operaciones:** Se trata de componentes informáticos para el proceso y presentación de la información.

Las funciones de una red de gestión se estructura en 4 niveles (es decir, cada tipo de gestión se realiza en estratos diferentes) de acuerdo con ITU-T M.3010:

- **Gestión de sistema BML:** (Business Management Layer) para modelos de largo plazo, planes de servicios y tarifas.
- **Gestión de servicio SML:** (Service Management Layer) para la administración de órdenes de servicio.
- **Gestión de red NML:** (Network Management Layer) para gestión de alarmas, tráfico, performance y configuración de la red.
- **Gestión de elemento de red EML:** (Element Management Layer) gestión de alarmas, tráfico, performance y configuración del equipo.
- **Gestión local del elemento de red NEL:** (Network Element Layer) para las funciones locales de gestión.

4.6.1 COMPONENTES DE LA GESTION SDH

Los componentes que integran la red de gestión SDH son los siguientes:

- Unidad de Control y unidad de Gestión del equipo.
- Canal de comunicación hacia la PC en terminal local.
- Canal de comunicación entre equipos de la misma red.
- Red de comunicación entre distintos equipos en una misma estación.
- Red de comunicación en el Centro de Gestión Regional.
- Red de comunicación entre Centros Regionales con el Centro Nacional Unificado.
- Software de aplicación.

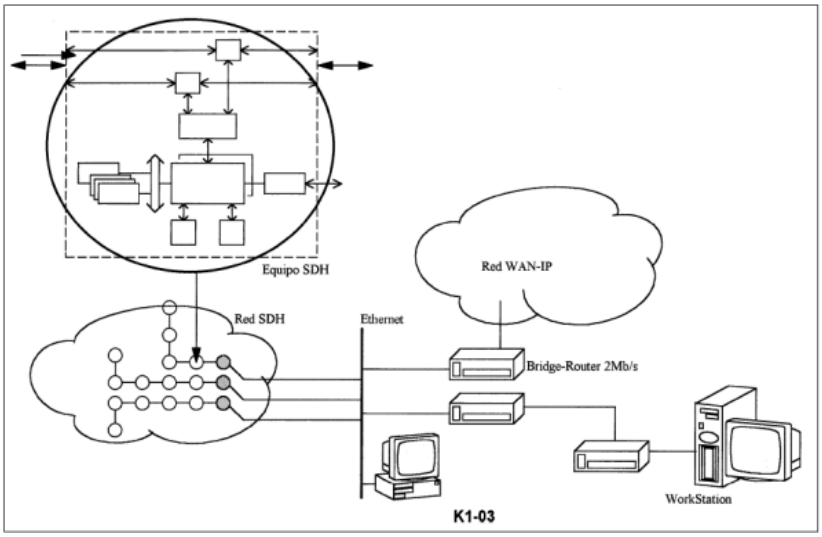


Fig. 4. 19. Estructura de enlace para gestión de equipos SDH [100]

4.6.1.1 Unidad de Control y Gestión

La unidad de control y gestión mantiene actualizada la base de datos del equipo y permite la comunicación con el operador del Terminal Local. Sus funciones en particular son:

- Comunicación con las distintas unidades del equipo. Se trata de un proceso de comunicación del tipo Polling donde la unidad de control interroga en forma periódica a las distintas unidades para actualizar la Base de Datos MIB.
- Actualización de la Base de Datos. En esta base de datos se sostiene la información de alarmas, configuración, reportes de performance, etc.
- Comunicación con la PC local. Permite realizar las operaciones de gestión local desde una PC mediante la interfaz F.
- Comunicación con la Unidad de Gestión de red TMN. Entre ambas unidades (Control y Gestión) se puede enlazar al equipo con la TMN.

100

http://www.rares.com.ar/albums/1_Manual_de_Telecomunicaciones_2000/407%20Gestion%20redes%20SDH.pdf

4.6.1.2 Canal de comunicación hacia la PC en la estación local

La interfaz F permite comunicar al equipo con una PC exterior de esta manera se puede realizar cambios en la programación local. Esta función es necesaria en la configuración inicial del equipo cuando aún no se han ingresado los parámetros de comunicación de red (direcciones MAC, NSAP e IP) que permiten la conexión remota.

4.6.1.3 Red de comunicación entre distintos equipos en una misma estación

En una estación pueden existir distintos tipos de equipos SDH y distintos enlaces que conforman la red. Para efectuar la interconexión de los mismos se requiere de la interfaz Q desde la Unidad de Gestión.

Interfaz Q1/Q2/Q3: Q1/Q2 se indica en la norma ITU-T G.771 y Q3 en la norma ITU-T G.773 que identifica las capas del modelo ISO. Existen 5 variantes para Q3 propuestas y denominadas A1/A2/B1/B2/B3. La variante Q3/B2 se usa para comunicación con protocolo X.25 mientras que la variante Q3/B3 se usa para una salida LAN Ethernet (la LAN pertenece al sistema de operación). En el caso de Q3/B3 se trata de la IEEE 802.2 para la red de área local LAN Ethernet.

Interfaz LAN ETHERNET: Normalmente los equipos SDH disponen de una interfaz física de conexión AUI que permite acceder al equipo mediante una LAN. Todos los equipos a ser gestionados por la TMN deben ser interconectados mediante esta LAN. El protocolo de capa 2 es el definido en IEEE 802.3 (MAC y LLC). Para configurar correctamente la LAN se debe programar a cada equipo con una dirección MAC distinta.

4.6.1.4 Red de comunicación en el Centro de Gestión Regional

En el Centro de Gestión Regional se concentra la gestión remota de los equipos en un sector de la red.

4.6.1.5 Red de comunicación entre Centros Regionales con el Centro Nacional Unificado

La capa física y de enlace de datos es la red LAN y WAN mediante routers. Utiliza los protocolos para las capas superiores. Permite la interconexión de varios Centros Regionales con el Centro Nacional, la interconexión se realiza mediante una red extensa conmutada por routers. El direccionamiento se efectúa mediante direcciones IP.

4.6.1.6 Software de Aplicación

El diseño e implementación del sistema de operaciones OS se basa en un software diseñado con la técnica orientada al objeto. En una red real la función completa envuelve la interacción de todos los objetos asociados. La totalidad de los objetos se la conoce como base o modelo de datos-información de gestión.

4.6 VENTAJAS Y DESVENTAJAS

Entre las ventajas que ofrece SDH se puede destacar las siguientes:

- Los sistemas modernos SDH alcanzan velocidades de 10 Gbps.
- SDH es la tecnología más adecuada para los backbones.
- Comparado con los tradicionales sistemas PDH, ahora es mucho más fácil extraer o insertar canales de menor velocidad en las señales compuestas SDH de alta velocidad, no hace falta demultiplexar y volver a multiplexar la señal.
- SDH permite a los proveedores de redes reaccionar fácil y rápidamente frente a las demandas de sus clientes.
- Las redes SDH incluyen mecanismos automáticos de protección y recuperación ante posibles fallas del sistema.
- SDH, hoy en día, es la plataforma ideal para multitud de servicios, desde la telefonía tradicional, las redes RDSI o la telefonía móvil hasta las

comunicaciones de datos (LAN, WAN, etc.) y es igual de adecuado para los servicios más recientes, como el video bajo demanda.

- Con SDH es más fácil intercalar entre los distintos proveedores de redes. Las interfaces SDH están normalizadas, lo que simplifica las combinaciones de elementos de redes de diferentes fabricantes.

Entre las desventajas de SDH se considera las siguientes:

- Se pierde eficiencia, ya que, el número de bytes destinados a la cabecera de sección es demasiado grande.
- Necesita sincronismo entre los nodos de la red, requiere que todos los servicios trabajen bajo una misma referencia de temporización.
- El entrelazamiento de bits hace que canales a 64 Kbps pertenecientes a un tramo de trafico solo se puedan separar hasta que se demultiplexa a nivel de multiplex primario.
- Los canales de n 64 Kbps que no se puedan incluir bajo el multiplex primario no se pueden tramitar de ninguna otra forma por la red.
- La información de mantenimiento no está asociada a vías completas de tráfico, sino a enlaces individuales, por lo cual el procedimiento de mantenimiento para una vía completa es complicado.

5. CAPITULO V

DESARROLLO DEL SOFTWARE DE SIMULACIÓN

5.1 ELECCIÓN DE LA HERRAMIENTA

Para la realización de esta simulación, se ha tomado en cuenta las ventajas y facilidades que ofrece la plataforma .Net, en especial su paquete C#, considerando lo siguiente.

C# es muy utilizado para aplicaciones de escritorio pero no para sistemas embebidos (como microcontroladores), sistemas en tiempo real ni kernels. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET el cual es similar al de Java aunque incluye mejoras derivadas de otros lenguaje

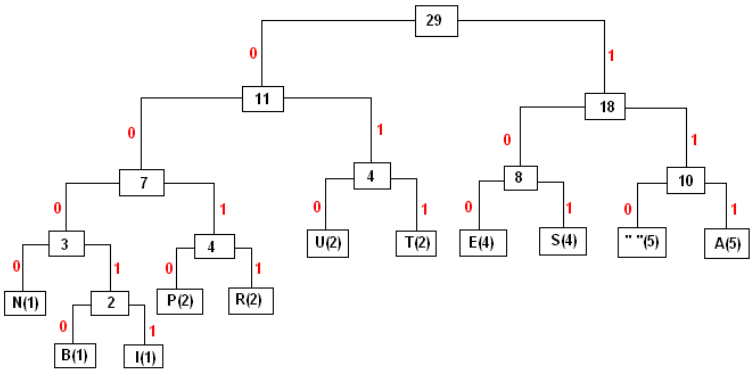
Por lo tanto, para C# crearon un lenguaje que pudiera ser entendido por programadores de C, C++ y Java. Si uno se acostumbra a programar en C#, se acostumbra a usar la sintaxis de los otros y viceversa, además líneas de código comunes como los if, for, try, etc. son iguales.

5.2 MODELO DEL ALGORITMO DE COMPRESIÓN

El modelo del algoritmo a presentar está basado en el algoritmo de compresión de Huffman (visto en el capítulo 2), el cual trabaja con una tabla de frecuencias donde se cuantifica las veces en que un caracter es repetido. Así, a los caracteres más repetidos se asigna un código de pocos bits y a los caracteres menos repetidos se les asigna un código más largo.

A continuación se crea un árbol binario tomando como referencia los valores más pequeños de la tabla de frecuencias, estos dos valores se suman y se juntarán a

un nuevo valor de la tabla, de esta manera se continúa hasta que el árbol tenga una sola raíz.



Para obtener un código de un caracter se asciende desde las hojas hacia la raíz. Si se asciende por la derecha se asigna 1, si es por la izquierda se asigna 0.

Con ello se obtiene la Tabla de Códigos.

CARACTER	FRECUENCIA	CÓDIGO
-	5	011
A	5	111
E	4	001
S	4	101
T	2	110
U	2	010
P	2	0100
R	2	1100
N	1	0000
B	1	01000
I	1	11000

Tabla 5.1 Tabla de códigos en base al algoritmo de codificación Huffman

Para codificar se sustituye el caracter por su respectivo código. Para el presente caso se tiene lo siguiente.

E	S	T	A	-	E	S	-	U	N	A	-	P	R	U	E	B	A	-
001	101	110	111	011	001	101	011	010	0000	111	011	0100	1100	010	001	01000	111	011

P	A	R	A	-	T	E	S	I	S
0100	111	1100	111	011	110	001	101	11000	101

Se agrupa los bits en bytes.

00110111	01110110	01101011	01101011	11011010	01100010	00101000	11101101	00111110	01110111	10001101	11000101

Para decodificar se debe contar con la tabla de códigos que se genera luego de armar el árbol y con los bytes del mensaje codificado.

Tabla de Códigos.

CARACTER	FRECUENCIA	CÓDIGO
-	5	011
A	5	111
E	4	001
S	4	101
T	2	110
U	2	010
P	2	0100
R	2	1100
N	1	0000
B	1	01000
I	1	11000

Tabla 5.2 Generación de códigos en base al algoritmo Huffman

Mensaje Codificado.

00110111	01110110	01101011	01101011	11011010	01100010	00101000	11101101	00111110	01110111	10001101	11000101

Con esto se tiene:

E	S	T	A	-	E	S	-	U	N	A	-	P	R	U	E	B	A	-
001	101	110	111	011	001	101	011	010	0000	111	011	0100	1100	010	001	01000	111	011

P	A	R	A	-	T	E	S	I	S
0100	111	1100	111	011	110	001	101	11000	101

Y nuevamente se tiene el mensaje original.

5.3 DISEÑO DEL ALGORITMO DE COMPRESION

Los pasos a seguir en el modelo del algoritmo planteado son los siguientes:

1. Elegir la Tecnología de Transmisión (E1, TU-12, AU-3, AU-4).
2. Ingresar las variables que intervienen en la comunicación (distancia, medio de transmisión).
3. Elegir el número de tramas a utilizar (ej. 1 E1, 3 E1's, etc.).
4. Elegir el número de usuarios.
5. Elegir el Ancho de Banda para cada usuario.
6. Elegir la información a transmitir.
7. Iniciar el proceso de compresión.
8. Verificar los resultados estadísticos.

Con estos antecedentes se detallan los diagramas UML.

5.3.1 DIAGRAMAS DE CASOS DE USO

Casos de Uso es una técnica para capturar información de cómo un sistema o negocio trabaja o desease que trabaje. No pertenece estrictamente al enfoque orientado a objeto, es una técnica para captura de requisitos.

Actores

- **Principales:** personas que usan el sistema.
- **Secundarios:** personas que mantienen o administran el sistema.
- **Otros sistemas:** sistemas con los que el sistema interactúa.

La misma persona física puede interpretar varios papeles como actores distintos, el nombre del actor describe el papel desempeñado.

Los Casos de Uso se determinan observando y precisando, actor por actor, las secuencias de interacción, los escenarios, desde el punto de vista del usuario.

Caso de Uso: Codificación Huffman
Actor principal: Operador/Usuario
Actor Secundario: Codificador/Compresor

Escenario principal de éxito:

1. El Operador/Usuario elige una Tecnología de Transmisión.
2. El Operador/Usuario agrega las Tramas de acuerdo a la Tecnología seleccionada anteriormente.
3. El Operador/Usuario agrega los usuarios finales.
4. El Operador/Usuario elige el Ancho de Banda asignado a cada usuario.
5. El Operador/Usuario elige los archivos (información) a enviar.
6. El Codificador/Compresor realiza la compresión de la información antes de ser enviada.
7. El Operador/Usuario visualiza los resultados obtenidos.

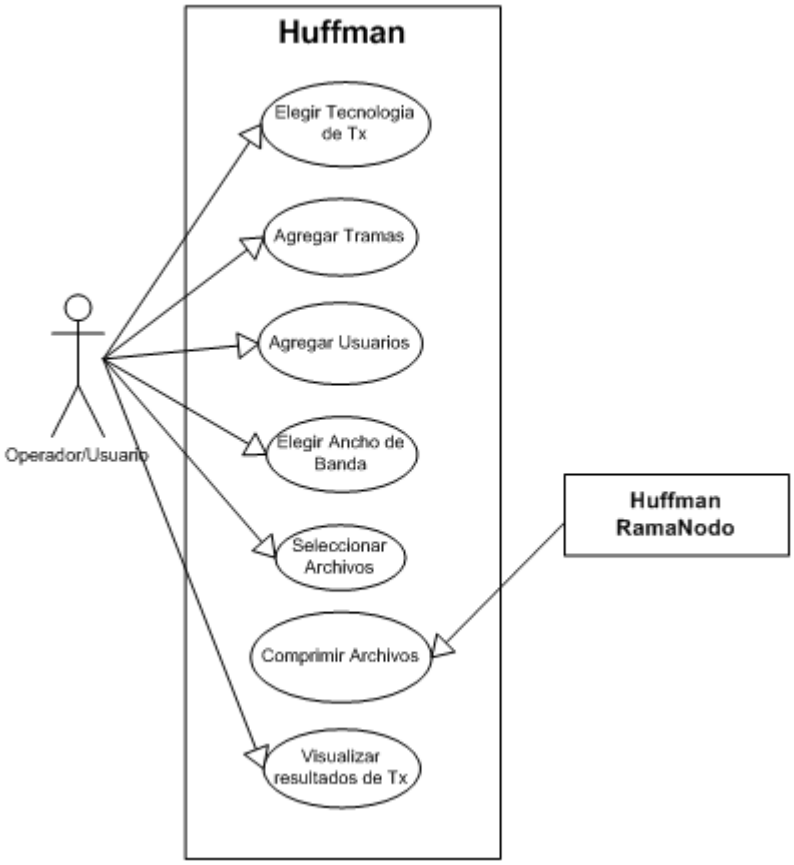


Fig. 5. 1 Diagrama de Casos de Uso

Caso de Uso: Elegir Tecnología de Tx.

Actor principal: Operador/Usuario

Actor Secundario: Sistema

Escenario principal de éxito:

1. El Operador/Usuario elige una Tecnología de Transmisión.
2. El Sistema devuelve el formulario de acuerdo al valor elegido.

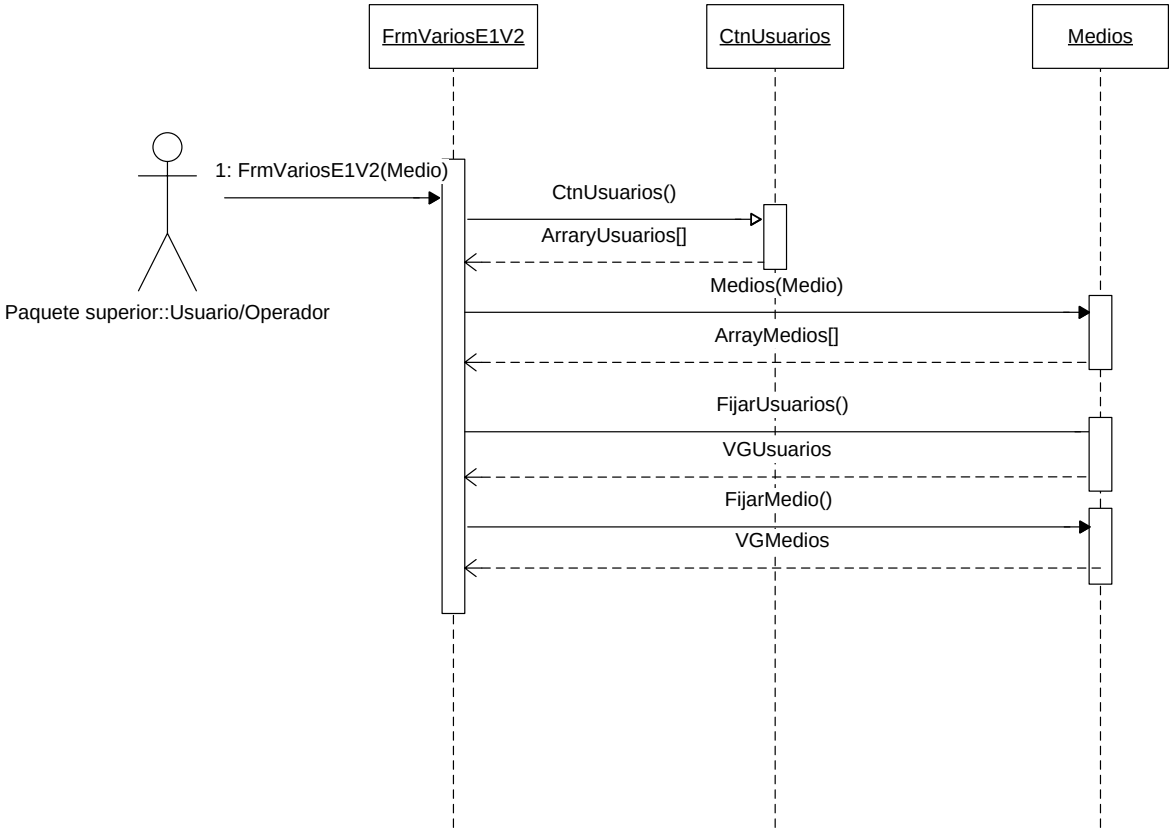


Fig. 5. 2 Diagrama de Secuencia Elegir Tecnología de Tx

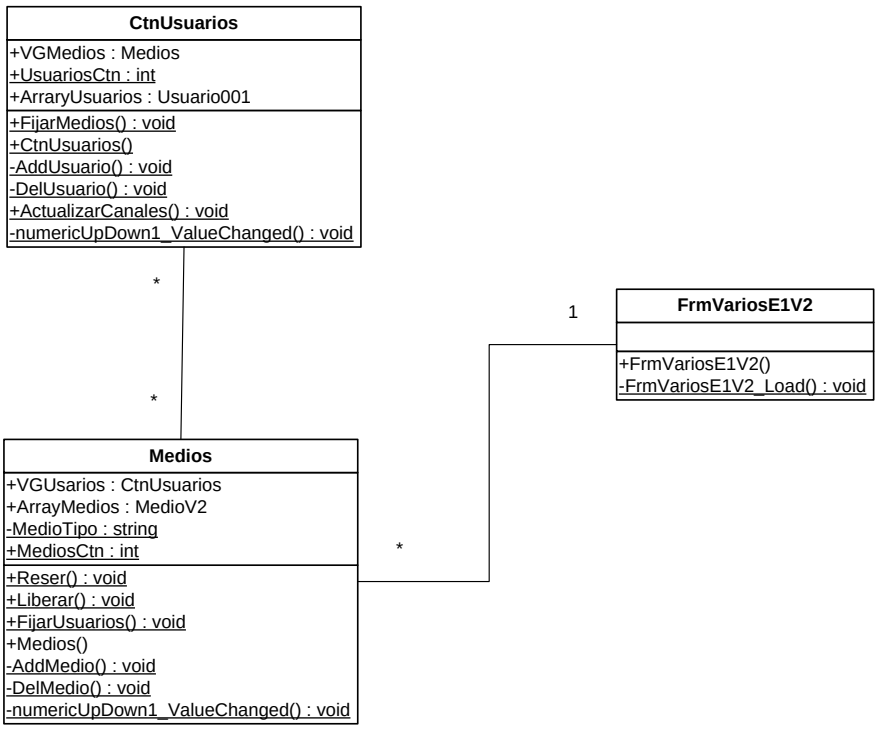


Fig. 5. 3 Diagrama de Clases elegir Tecnología de Tx

Caso de Uso: Agregar Tramas

Actor principal: Operador/Usuario

Actor Secundario: Sistema

Escenario principal de éxito:

1. El Operador/Usuario elige el número de tramas de acuerdo a la Tecnología de Transmisión seleccionada.
2. El Sistema devuelve el número de tramas seleccionadas por el usuario.

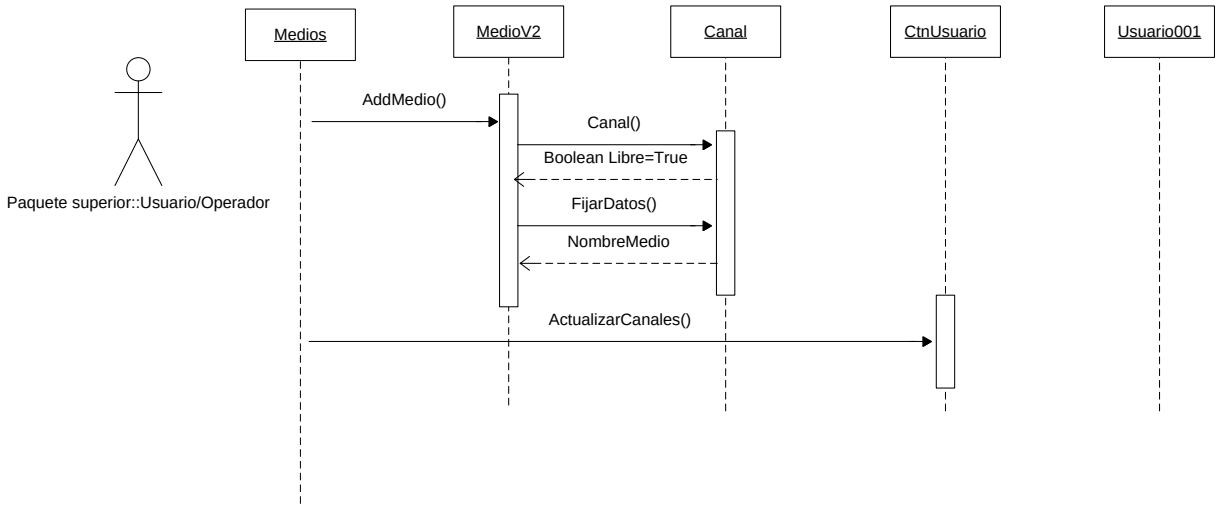


Fig. 5. 4 Diagrama de Secuencia Agregar Tramas

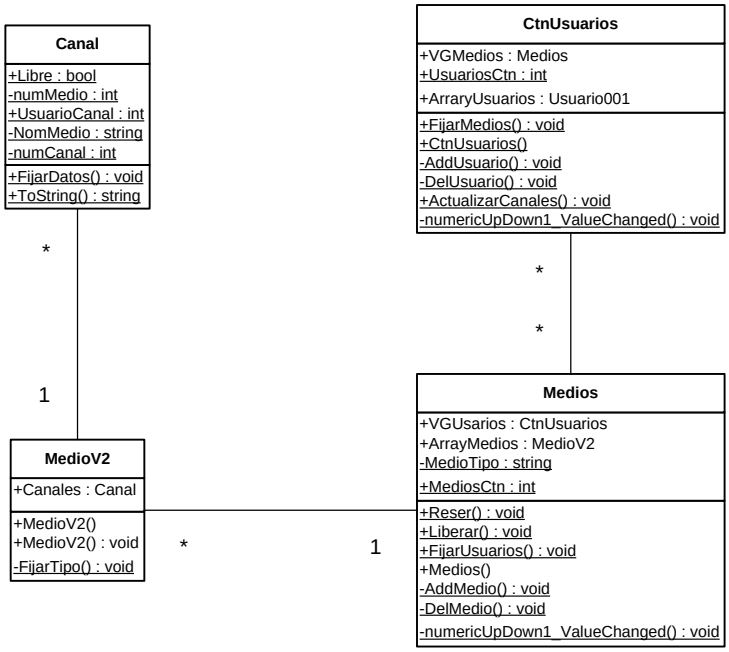


Fig. 5. 5 Diagrama de Clases Agregar Tramas

Caso de Uso: Agregar Usuarios
Actor principal: Operador/Usuario
Actor Secundario: Sistema

Escenario principal de éxito:

1. El Operador/Usuario elige el número de usuarios.
2. El Sistema devuelve la interfaz de usuario de acuerdo al valor ingresado por el usuario.

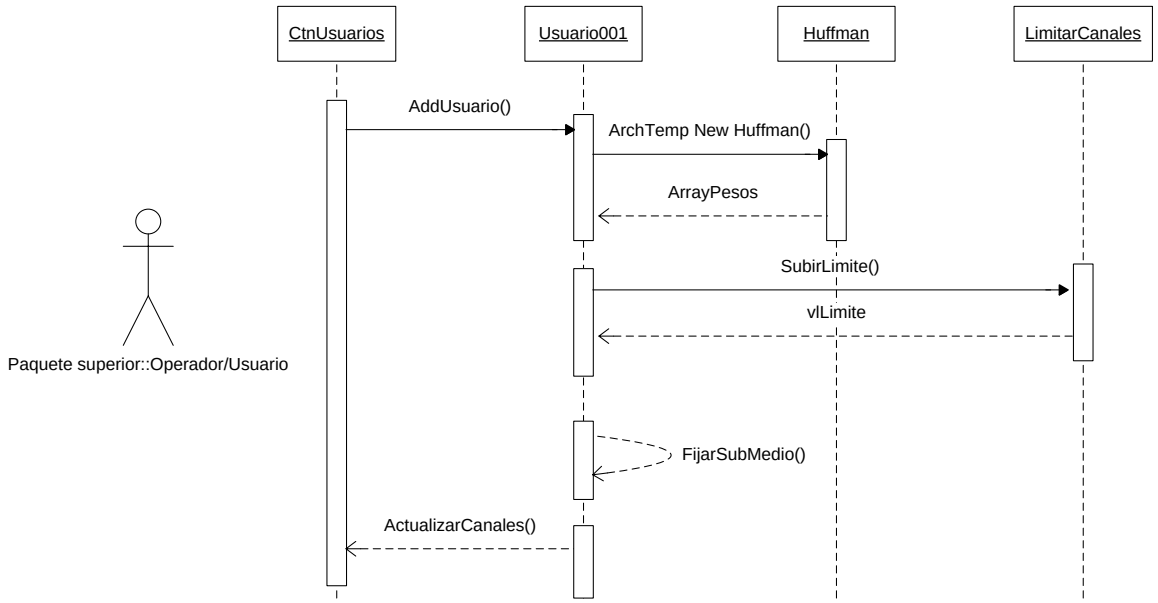


Fig. 5. 6 Diagrama de Secuencia Agregar Usuarios

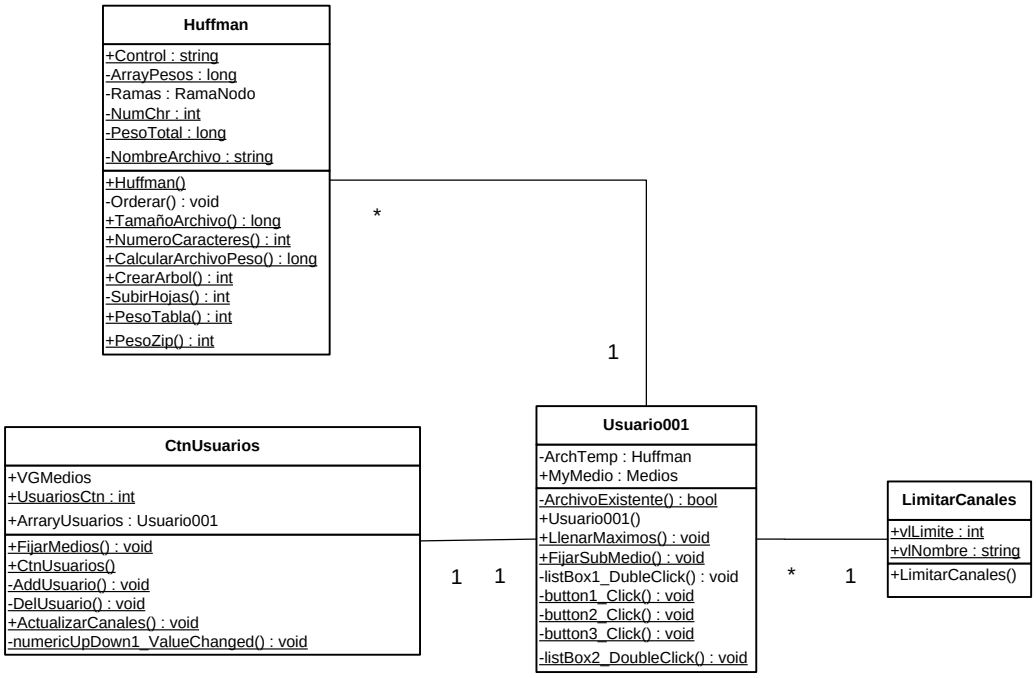


Fig. 5. 7 Diagrama de Clases Agregar Usuarios

Caso de Uso: Agregar Canales

Actor principal: Operador/Usuario

Actor Secundario: Sistema

Escenario principal de éxito:

1. El Operador/Usuario elige el número de canales (Ancho de Banda).
2. El Sistema devuelve el Ancho de Banda en la interfaz de usuario de acuerdo al valor elegido por el usuario.

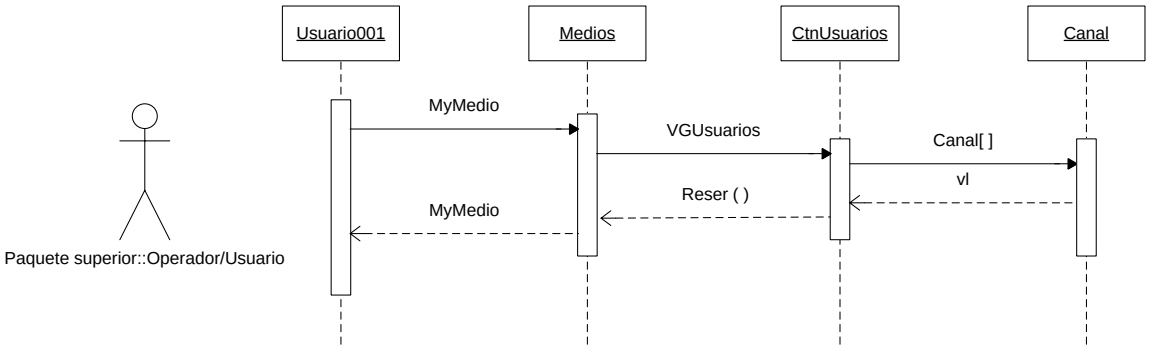


Fig. 5. 8 Diagrama de Secuencia Agregar Canales

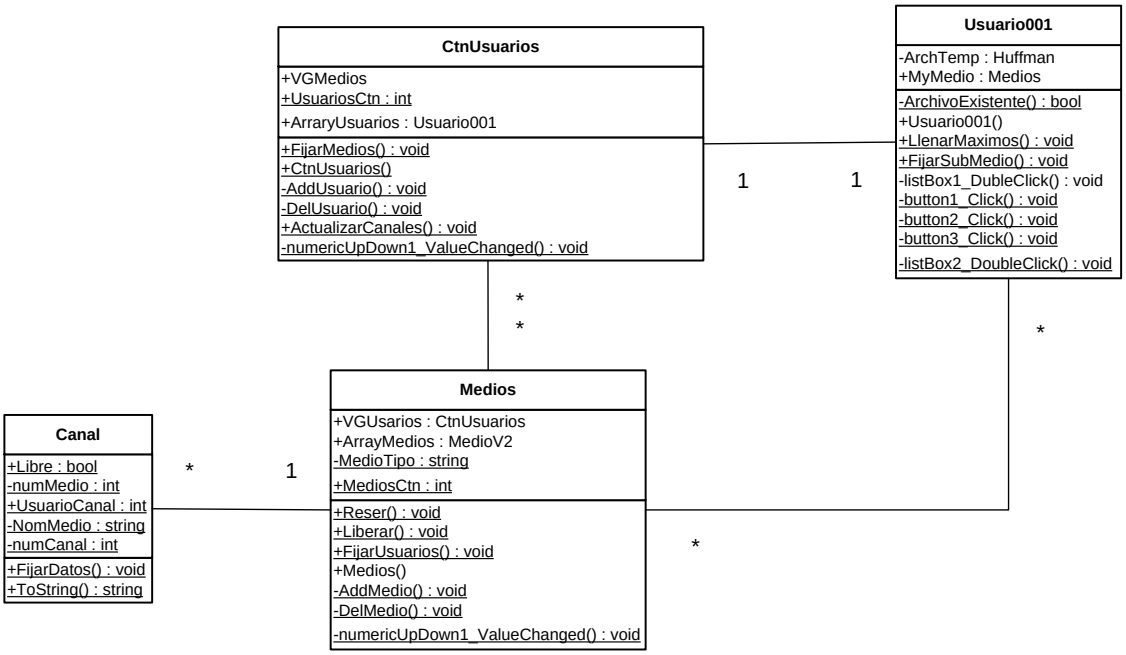


Fig. 5. 9 Diagrama de Clases Agregar Canales

Caso de Uso: Codificar
Actor principal: Operador/Usuario
Actor Secundario: Sistema

Escenario principal de éxito:

1. El Operador/Usuario elige los archivos a transmitir.
2. El Sistema devuelve los cálculos estadísticos de acuerdo a la cantidad de información transmitida.

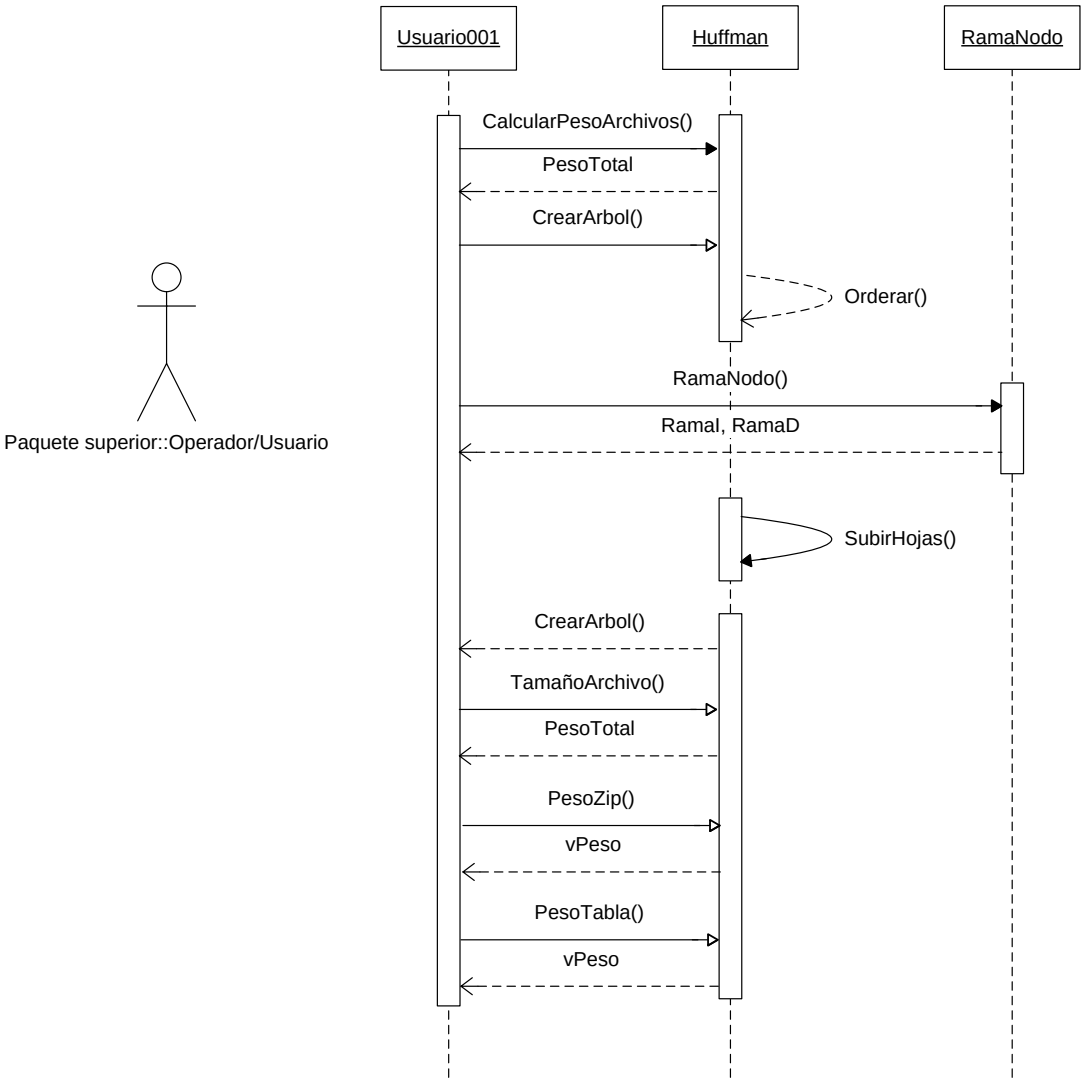


Fig. 5. 10 Diagrama de Secuencia Codificar

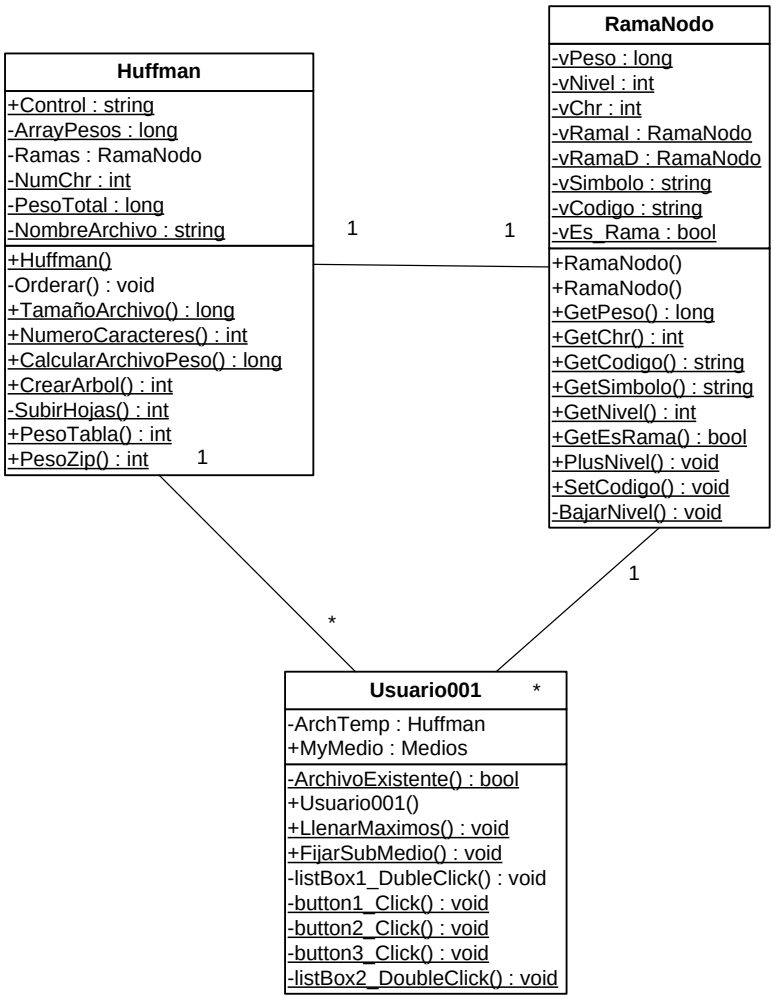


Fig. 5. 11 Diagrama de Clases Codificar Huffman

5.3.2 DIAGRAMA DE MODELO DE DOMINIO

Los modelos de dominio se utilizan para expresar el entendimiento de un análisis previo al diseño de un sistema, ya sea de software o de otro tipo.

Similares a los mapas mentales utilizados en el aprendizaje, el modelo de dominio es utilizado por el analista como un medio para comprender el sector industrial o de negocios al cual el sistema va a servir.

En el presente diseño se considera el siguiente modelo de Dominio.

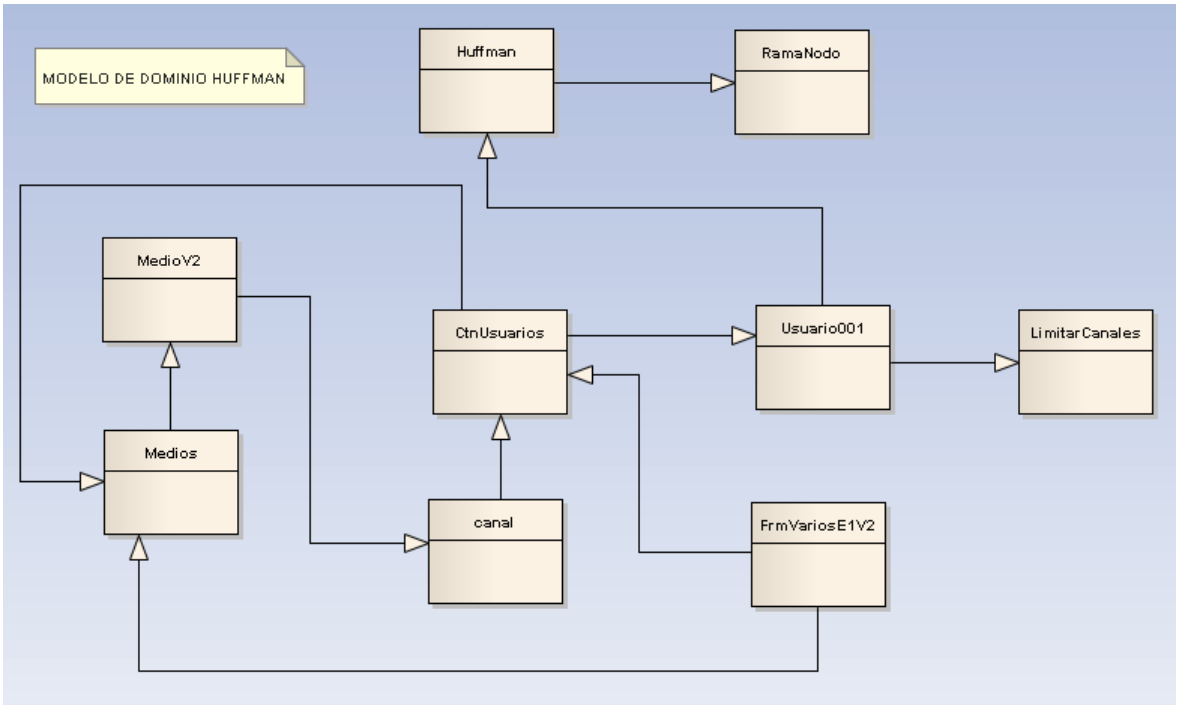


Fig. 5. 12 Diagrama de Modelo de Dominio

El Formulario Principal (FrmVariosE1V2) contiene dos paneles para agregar usuarios (CtnUsuarios) y tramas (Medios) de acuerdo al tipo de tecnología utilizada en la transmisión, dentro de las tramas se incluyen los canales (Canal) que forman la trama seleccionada .

La parte de los usuarios está definida por una interfaz de usuario (Usuario001) el cual controla los posibles anchos de banda a elegir (LimitarCanales).

La parte de la codificación de la información del usuario (Usuario001) se encarga el algoritmo (Huffman) junto con RamaNodo.

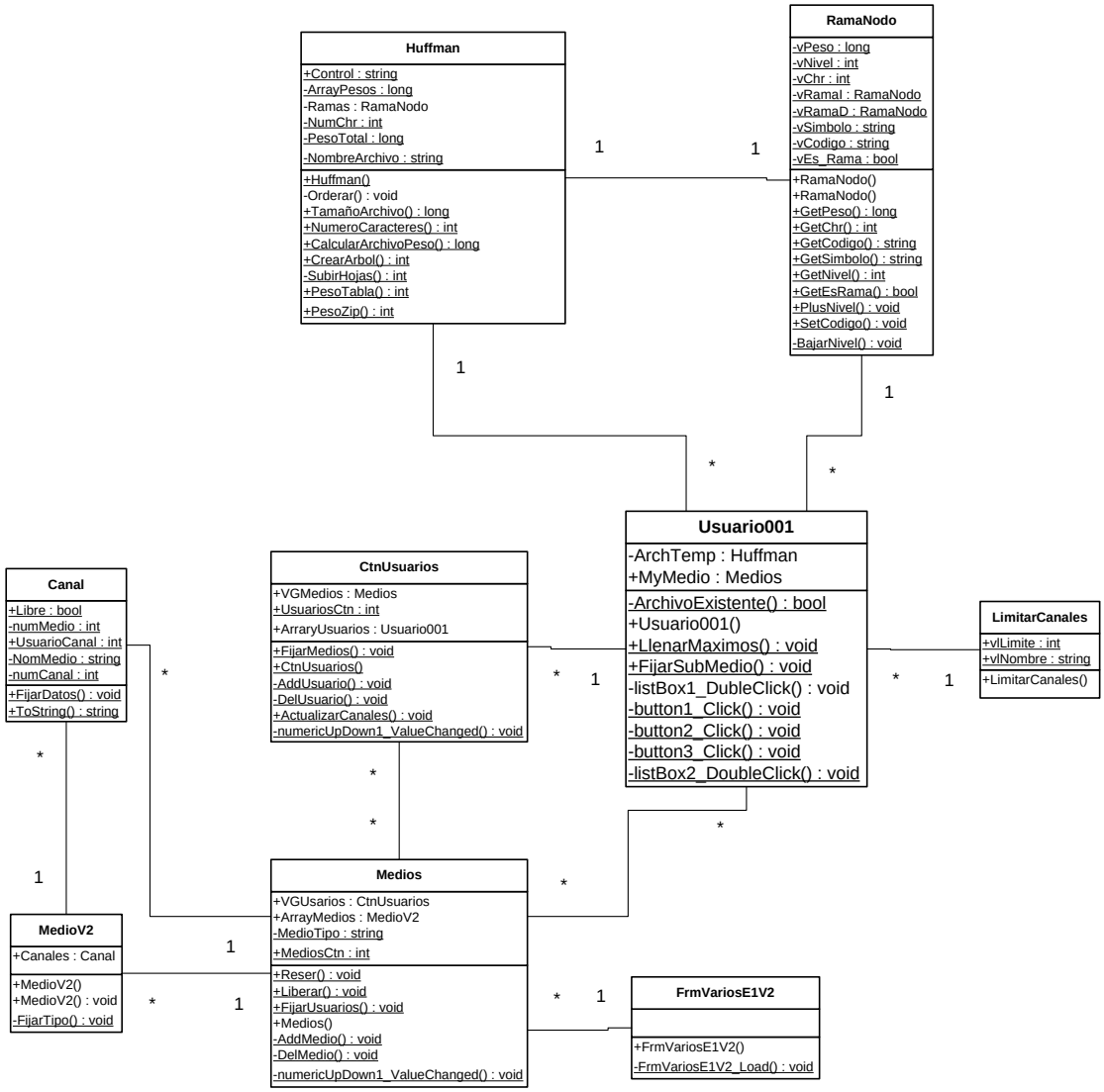


Fig. 5. 13 Diagrama de Clases

5.3.3 DICCIONARIO DE DATOS

A continuación, se figura el diagrama de clases correspondiente al Sistema Codificación Huffman, en donde se visualizará las clases y operaciones que se utilizan para el correcto funcionamiento del software.

DICCIONARIO DE DATOS – DESCRIPCIÓN DE ENTIDADES Y ATRIBUTOS

CLASE MDIParent: Formulario Principal que contiene los menús que acceden a los diferentes formularios.

Fig. 5. 14. Formulario Principal

Descripción de menús:

Menú Simulaciones	Función asociada	Descripción
Archivo en E1	arcivoEnE1ToolStripMenuItem_Click	Crea un nuevo formulario en el cual se puede ver cómo se ordena el contenido de un archivo en los canales de un E1
Varios Usuarios un E1	variosUsuariosToolStripMenuItem_Click	Crea un formulario en el cual se ven simulaciones de compresión y tiempos de transmisión en un solo medio E1 y con varios usuarios
Varios Usuarios varios E1	pruebaV2ToolStripMenuItem_Click	Crea un nuevo formulario que es una variación del anterior, este también trabaja con varios usuarios pero permite trabajar con varios E1
Varios Usuarios STM1 -> TU – 12	tU12ToolStripMenuItem_Click	Formulario que permite crear simulaciones de compresión con varios usuarios y utiliza contenedores TU-12
Varios Usuarios STM1 -> VC – 3	vCToolStripMenuItem_Click	Formulario que permite crear simulaciones con varios usuarios y utiliza contenedores VC-3
Varios Usuarios STM1 -> VC – 4	vC4ToolStripMenuItem_Click	Formulario que permite crear simulaciones con varios usuarios y utiliza contenedoresVC-4

Tabla 5.3 Menús del Formulario Principal

CLASE Form1: Formulario que permite visualizar la asignación de información en una Trama E1.

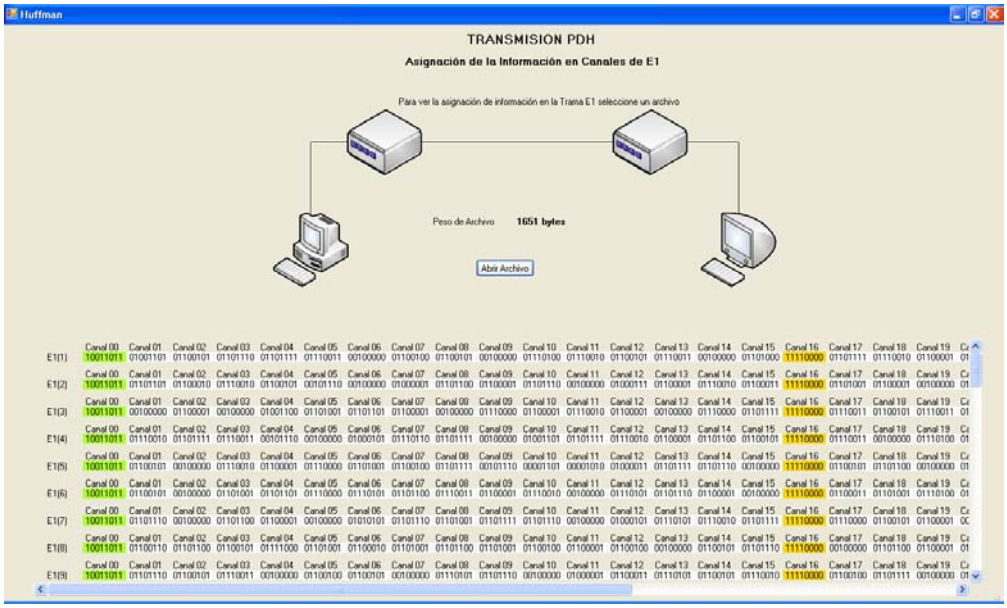


Fig. 5. 15. Asignación de la información en una trama E1

Variables:

Variable	Tipo	Descripción
E1	Array	Matriz de canales
ArchivoPeso	Int	Control de peso de archivo
x, y, l	Int	Variables para ubicación de los valores en la matriz.

Tabla 5.4 Variables de la Clase Form1

Funciones:

Función	Tipo devuelto	Descripción
AgregarValor	Void	Ubica el valor del caracter en binario y también se encarga de crear dinámicamente los canales
Cabeza	string	Retorna el valor reservado de los canales de control 0 y 16
IntToBinary	String	Transforma el caracter en su equivalente a binario
IntToBinary8	String	Utiliza la función IntToBinary y completa el byte a 8 bits
button1_Click	Void	Función encargada de cargar los archivos

Tabla 5.5 Funciones de la Clase Form1

CLASE Form2: Formulario que permite visualizar la simulación de compresión y envió de la información proveniente de varios usuarios sobre una sola Trama E1.

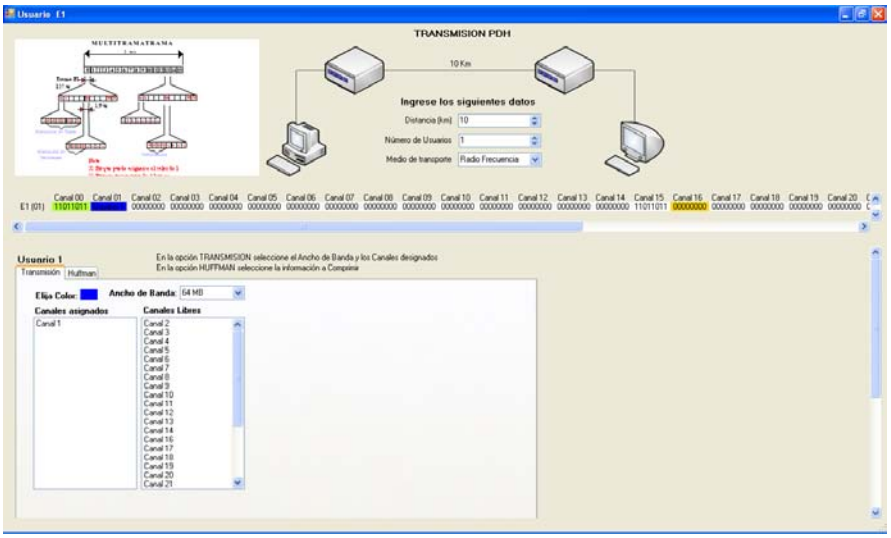


Fig. 5. 16 Interfaz de varios usuarios interactuando con una trama E1

Variables:

Variable	Tipo	Descripción
ListaUsuarios	UserE1	Matriz de usuarios que usan el canal.
TotalUsuarios	Int	Variable de control de número de usuarios
FileFlash	String	Devuelve la ubicación de las animaciones en formato Flash, en este caso animaciones de un E1

Tabla 5.6 Variables de la Clase Form2

Funciones:

Función	Tipo devuelto	Descripción
ActualizarArreglo	Void	Función encargada de actualizar el contenido de cada canal del medio de transmisión.
Form2_Load	Void	Función de iniciación de variables
numericUpDown2_ValueChanged	Void	Función encargada de controlar el número de usuarios

Tabla 5.7 Funciones de la Clase Form2

Nota: El objeto UserE1 es el encargado de realizar toda la codificación de los archivos de esta pantalla.

CLASE FrmVariosE1V2: Formulario que permite visualizar la simulación de compresión y envió de la información proveniente de varios usuarios sobre varias Tramas E1, VC-3 y VC-4 dependiendo de la tecnología de transmisión seleccionada en el menú de la pantalla principal.

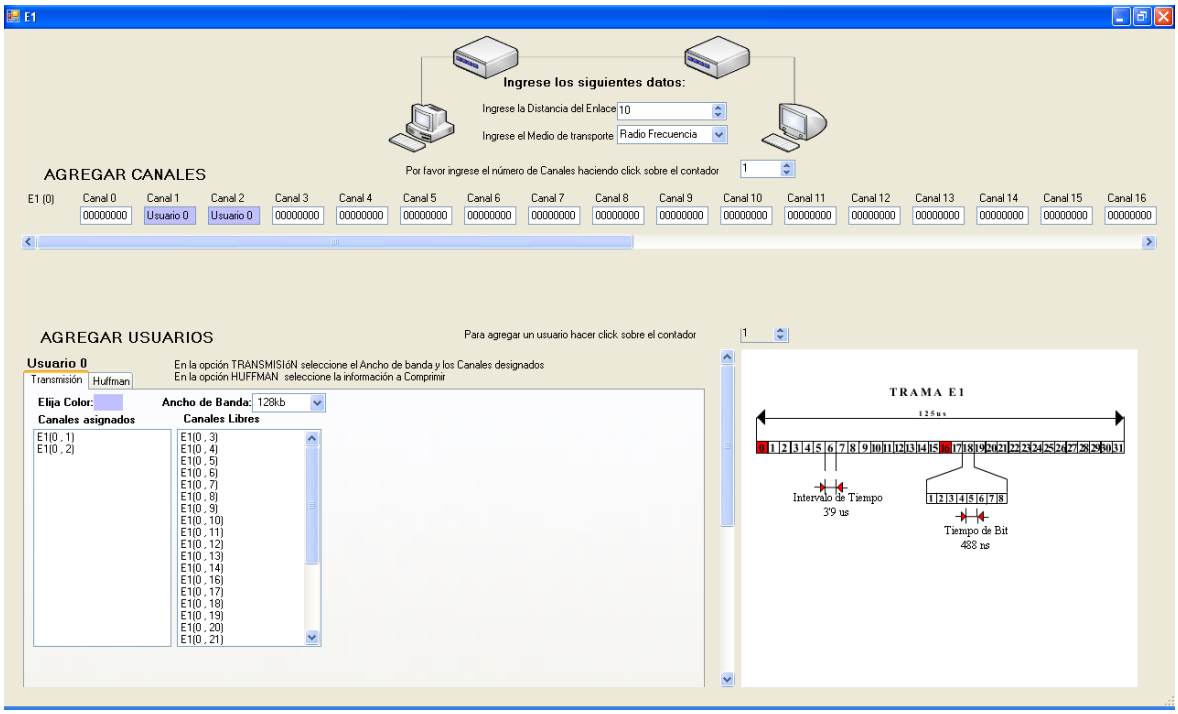


Fig. 5. 17 Interfaz de varios usuarios interactuando con varios E1

Variables:

Variable	Tipo	Descripción
ctnUsuarios1	CtnUsuarios	Contenedor de usuarios genérico, en este caso para E1, VC3 y VC4
medios1	Medios	Contenedor de medios genérico, en este caso para E1, VC3 y VC4

Tabla 5.8 Variables de la Clase FrmVariosE1V2

Funciones:

Función	Tipo devuelto	Descripción
FrmVariosE1V2_Load	Void	Esta es la función principal en la cual se relacionan los contenedores de usuarios y de medios. Una vez asociados estos contenedores actúan para mostrar los datos.

Tabla 5.9 Funciones de la Clase FrmVariosE1V2

CLASE STM1_TU12: Formulario que permite visualizar la simulación de compresión y envió de la información proveniente de varios usuarios sobre varias Tramas STM-1 en canales de Unidades Tributarias de órden 12 TU-12

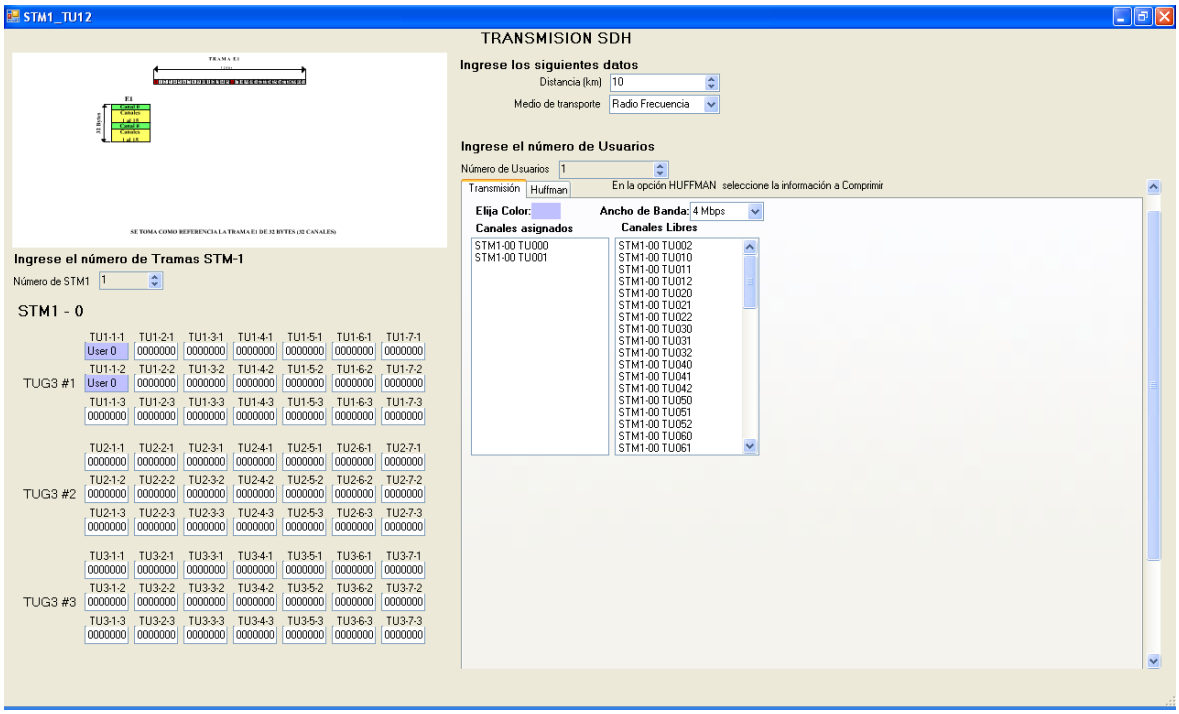


Fig. 5. 18 Interfaz de varios usuarios interactuando con una trama STM-1 a nivel de TU-12

Variables:

Variable	Tipo	Descripción
ctnTU121	CtnTU12	Contenedor de medios TU12
ctnUserTU121	CtnUserTU12	Contenedor de usuarios para medios TU12

Tabla 5.10 Variables de la Clase STM_TU12

Funciones:

Función	Tipo devuelto	Descripción
STM1_TU12	Constructor	Esta es la función principal en la cual se relacionan los contenedores de usuarios y de medios. Una vez asociados estos contenedores actúan para mostrar los datos.

Tabla 5.11 Funciones de la Clase STM_TU12

CLASE Canal: Control de usuario que permite llenar una Trama con el número de canales correspondiente a cada Trama.

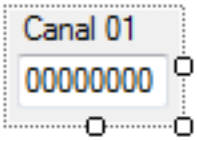


Fig. 5. 19 Canal de una trama

Variables:

Variable	Tipo	Descripción
Libre	Boolean	Esta variable permite saber si el canal esta libre
numMedio	Int	Variable que almacena a qué medio pertenecen los canales
UsuarioCanal	Int	En el caso de que el canal este ocupado se almacena el identificador del usuario
numCanal	Int	Identifica el número de canal en el medio actual
NomMedio	String	Almacena el nombre del medio

Tabla 5.12 Variables Clase Canal

Funciones:

Función	Tipo devuelto	Descripción
FijarDatos	Void	Almacena las variables de nombre del medio, número del medio y número del canal
ToString	String	Retorna el nombre del canal que es utilizado por el medio

Tabla 5.13 Funciones de la Clase Canal

CLASE CtnTU12: Control de usuario que almacena objetos TU12 permite llenar una Trama con el número de canales correspondiente a cada Trama.

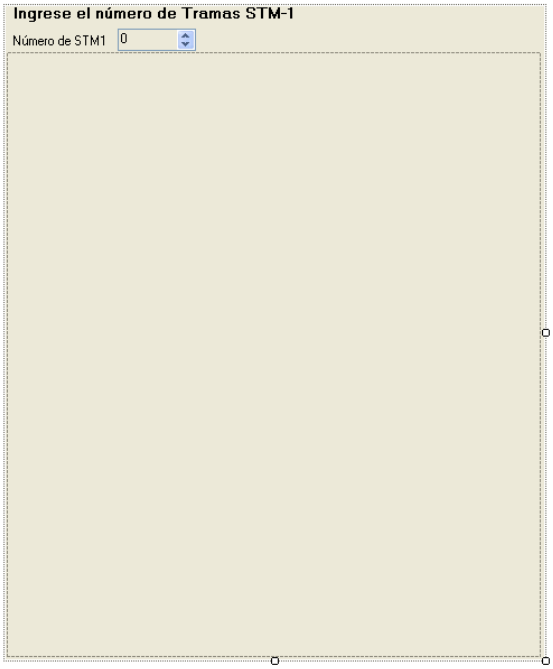


Fig. 5. 20 Contenedor de tramas STM-1 a nivel de TU-12

Variables:

Variable	Tipo	Descripción
cuentaTu	Int	Cantidad de medios TU12
ArrayTU12	TU_12	Arreglo contenedor de medios TU12
UsuariosTU12	CtnUserTU12	Objeto contenedor de usuarios TU12 al que hace referencia los medios

Tabla 5.14 Variables de la Clase CtnTU12

Funciones:

Función	Tipo devuelto	Descripción
fijarUsuarios	Void	Función que permite asociar el objeto UsuariosTU12.
AddStm1	Void	Función para agregar canales STM1
DelStm1	Void	Función encargada de borrar canales STM1
FijarCanal	Void	Permite asignar un canal a un usuario especifico
LiberarCanal	Void	Libera un canal especifico para que pueda ser utilizado por todos los usuarios

Tabla 5.15 Funciones de la Clase CntTU12

CLASE CtnUserTU12: Control de usuario que almacena objetos de usuarios TU12. .

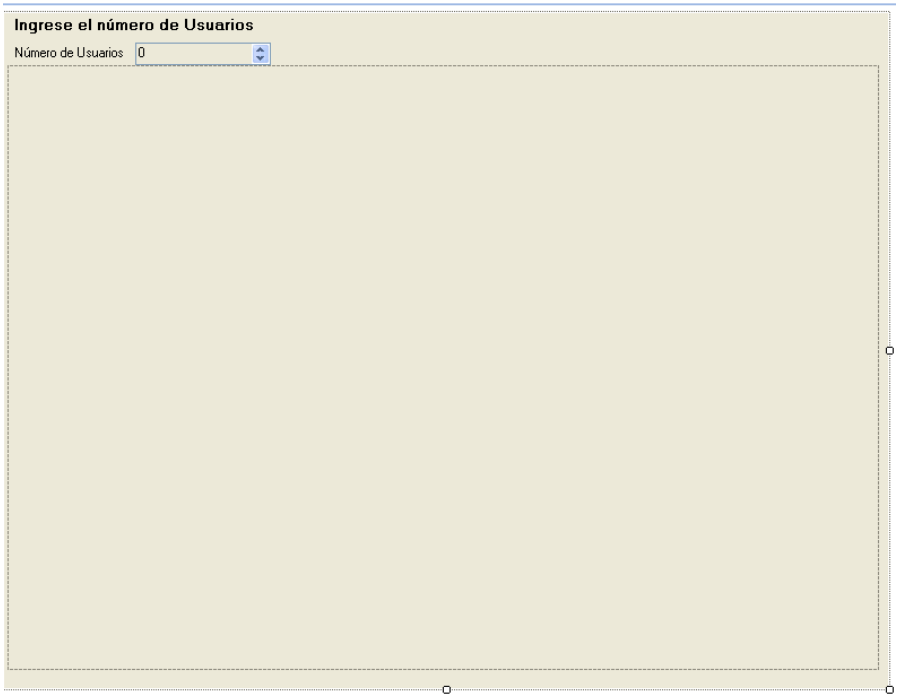


Fig. 5. 21 Interfaz del contenedor de usuarios que interactúan con tramas STM-1 a nivel de TU-12

Variables Importantes:

Variable	Tipo	Descripción
VGMedios	CtnTU12	Objeto contenedor de medios al que se hace referencia
UsuariosCtn	Int	Número de usuarios
ArraryUsuarios	Usuario001	Arreglo de usuarios de los canales TU12

Tabla 5.16 Variables de la Clase CtnUserTU12

Funciones:

Función	Tipo devuelto	Descripción
AddUsuario	Void	Función encargada de añadir nuevos usuarios
DelUsuario	Void	Función encargada de quitar usuarios
FijarMedio	Void	Esta función permite asociar el control de usuario con un contenedor de canales
ActualizarCanales	Void	Esta función es la encargada de actualizar los canales dependiendo del usuario al que han sido asignados

Tabla 5.17 Funciones de la Clase CtnUserTU12

CLASE CtnUsuarios: Control de usuario que almacena objetos de usuarios normales.

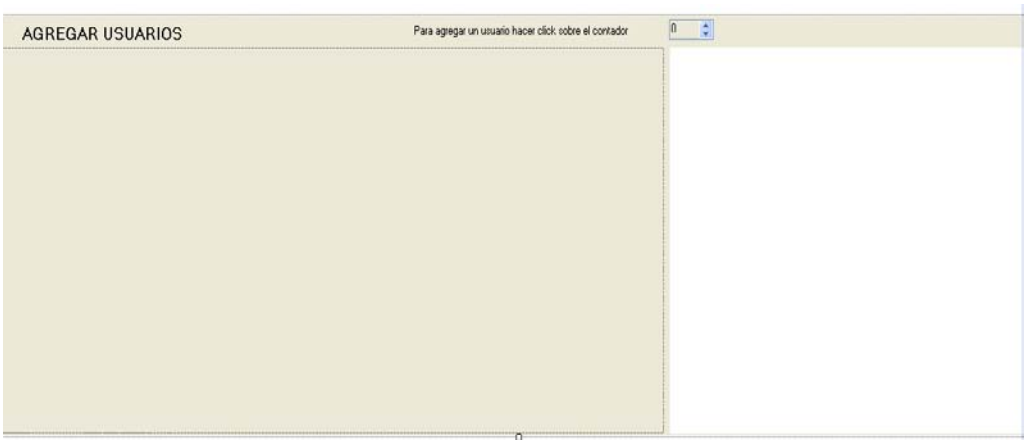


Fig. 5. 22 Interfaz de usuarios que interactúan con tramas E1, STM-1 a nivel VC-3 y VC-4

Variables:

Variable	Tipo	Descripción
FileFlash	String	Devuelve la ubicación de las animaciones en formato Flash, en este caso animaciones de un E1, STM1-AU3 y STM1-AU4, dependiendo de la tecnología seleccionada
VGMedios	Medios	Objeto que hace referencia a los medios asociados a los usuarios
UsuariosCtn	Int	Contador de usuarios
ArraryUsuarios	Usuario001	Arreglo que contiene el total de usuarios

Tabla 5.18 Variables de la Clase CtnUsuarios

Funciones:

Función	Tipo devuelto	Descripción
FijarMedios	Void	Esta función permite establecer el objeto Medios que está asociado.
AddUsuario	Void	Función que permite incrementar el número de los usuarios
DelUsuario	Void	Función que permite disminuir el número de usuarios
ActualizarCanales	Void	Permite actualizar el contenido del control Medios asociado, esta función actualiza en contenido de los canales según los usuarios

Tabla 5.19 Funciones de la Clase CtnUsuarios

CLASE E1: Control de usuario que contiene la estructura de una Trama E1.

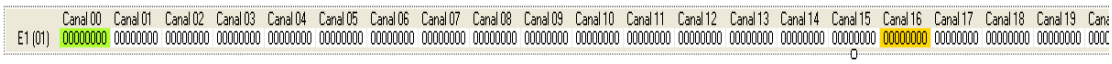


Fig. 5. 23 Control que contiene la estructura de la trama E1

Variables:

Variable	Tipo	Descripción
V2	Boolean	Permite saber si el control es utilizado en la versión de un E1 o de varios E1
Canales	Label	Es un arreglo de controles “Label” los cuales almacenan información de los canales del E1

Tabla 5.20 Variables de la Clase E1

Funciones:

Función	Tipo devuelto	Descripción
Asociar	Void	Permite asociar cada uno de los controles “Label” con el arreglo de controles del E1
SetV2	Void	Esta función ayuda a fijar el valor de V2
SetValue	Void	Permite fijar el usuario que usa un canal predeterminado
SetColor	Void	Cambia el color de fondo del canal elegido por el usuario

Tabla 5.21 Funciones de la Clase E1

CLASE Medios: Control de usuario que funciona como contenedor y permite almacenar un tipo de Trama con sus respectivos canales.

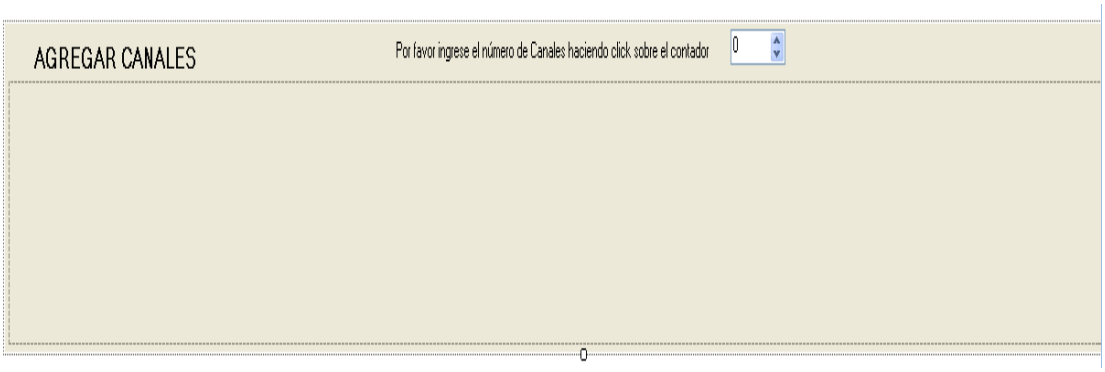


Fig. 5. 24 Interfaz del contenedor de tramas

Variables:

Variable	Tipo	Descripción
VGUsuarios	CtnUsuarios	Es el objeto que hace referencia al contenedor de usuarios
ArrayMedios	MedioV2	Arreglo que controla los objetos de medios
MediosCtn	Int	Número de medios que tiene el contenedor
MedioTipo	String	Ya que este es un contenedor genérico, esta variable almacena el nombre del tipo de medio que se va a utilizar

Tabla 5.22 Variables de la Clase Medios

Funciones:

Función	Tipo devuelto	Descripción
Reser	Void	Esta función permite asociar un canal con un usuario específico
Liberar	Void	Esta función se utiliza para liberar un canal y dejarlo utilizable para cualquier usuario
FijarUsuarios	Void	Esta es la función encargada de asociar el contenedor de usuarios con el contenedor de medios.
AddMedio	Void	Función por medio de la cual se puede añadir nuevos Medios al contenedor
DelMedio	Void	Función por medio de la cual se puede quitar Medios del contenedor

Tabla 5.23 Funciones de la Clase Medios

CLASE MediosV2: Control de usuario que contiene canales de una Trama, es completamente configurable y se ajusta a las necesidades de la simulación. Presenta los canales configurados en forma horizontal.

Fig. 5. 25 Interfaz del contenedor de tramas E1

Variables:

Variable	Tipo	Descripción
Canales	Canal	Es el arreglo de canales que contiene el Medio

Tabla 5.24 Variables de la Clase MediosV2

Funciones:

Función	Tipo devuelto	Descripción
MedioV2	Constructor	Es el encargado de configurar el nombre del medio, el número de canales y la velocidad de cada canal

Tabla 5.25 Funciones de la Clase MediosV2

CLASE TU-12: Control de usuario diseñado para una mejor visualización de los canales TU-12 de una Trama STM-1

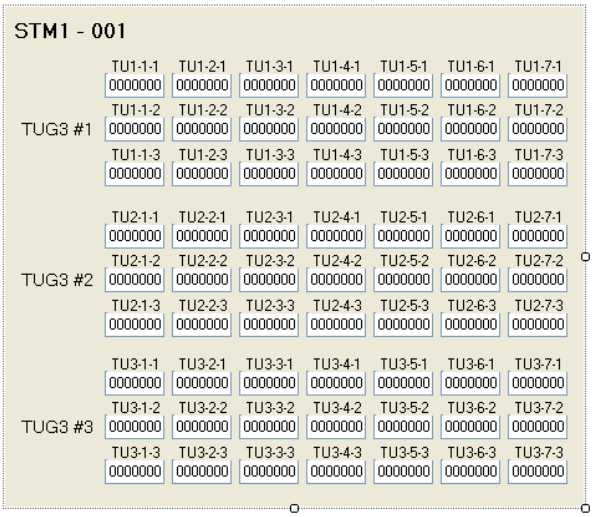


Fig. 5. 26 Interfaz de la agrupación de canales TU12

Variables:

Variable	Tipo	Descripción
ArrayText	TextBox	Arreglo de “Label” de tres posiciones para controlar la asignación de usuarios

Tabla 5.26 Variables de la Clase TU-12

Funciones:

Función	Tipo devuelto	Descripción
AsociarCanales	Void	Esta función es la encargada de asociar los “Labels” que representan los canales con los punteros del arreglo

Tabla 5.27 Funciones de la Clase TU-12

CLASE UserE1: Control de usuario encargado de mostrar la interfaz del usuario donde se puede seleccionar el Ancho de Banda, los canales asignados al usuario, la información a codificar y las estadísticas de la simulación. Trabaja con una sola Trama.

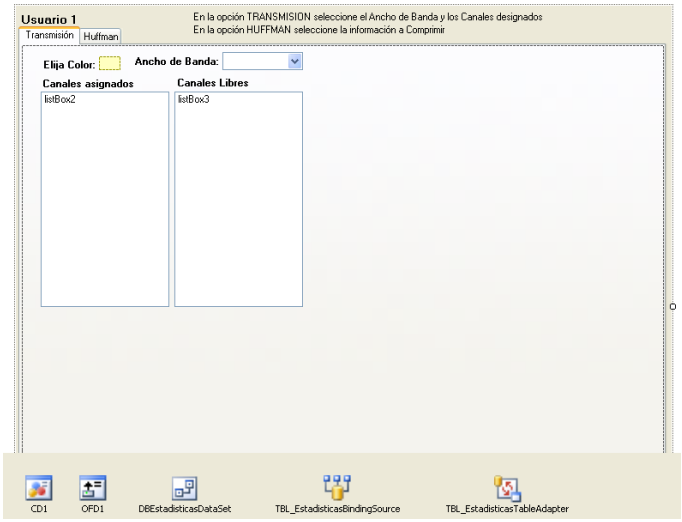


Fig. 5. 27 Interfaz de usuario

Variables:

Variable	Tipo	Descripción
MyE1	E1	Control de usuario que contiene los canales para la simulación del E1
ArchTemp	Huffman	Objeto de la clase Huffman, este es el encargado de la codificación, cálculos de tiempos y almacenamiento de resultados para su futura presentación

Tabla 5.28 Variables de la Clase UserE1

Funciones:

Función	Tipo devuelto	Descripción
ArchivoExiste	Bool	Función encargada de verificar si el archivo que el usuario especificó existe o no
LlenarCanales	Void	Esta función llena un ComboBox que contiene la velocidad de los canales por ejemplo 64 Kbps
ActualizarCanales	Void	Es la función encargada de inicializar los canales de la trama E1 que se va a utilizar, también reserva la ubicación 0 y 16 que son canales de control
LiberarCanales	Void	Esta función es la encargada de liberar los canales en el caso de que dejen de existir usuarios

Tabla 5.29 Funciones de la Clase UserE1

CLASE Usuario001: Control de usuario encargado de mostrar la interfaz del usuario donde se puede seleccionar el Ancho de Banda, los canales asignados al usuario, la información a codificar y las estadísticas de la simulación. Trabaja con varias Tramas.

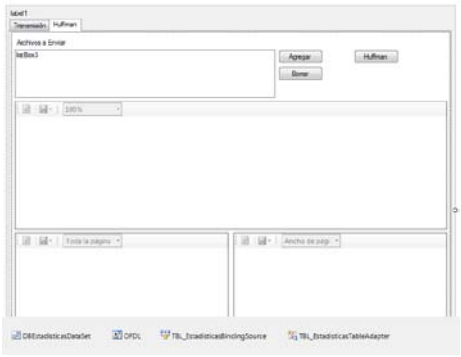


Fig. 5. 28 Interfaz de usuario para selección de Ancho de Banda

Variables:

Variable	Tipo devuelto	Descripción
Velocidad	Int32	Debido a que este control puede trabajar con diversos tipos de medios
MyMedio	Medios	Es la variable que permite relacionar este control con un contenedor de medios en todos los casos que no sean TU12
MyMedioTU12	CtnTU12	Es la variable que permite relacionar este control con un contenedor de medios siempre y cuando los medios sean TU12
ArchTemp	Huffman	Objeto de la clase Huffman, este es el encargado de la codificación, cálculos de tiempos y almacenamiento de resultados para su futura presentación
Limites	LimitarCanales	Esta variable es la encargada de controlar el límite de canales y la velocidad de los mismos
NumMax	int	Es el número de usuarios que controla la aplicación

Tabla 5.30 Variables de la Clase Usuario001

Funciones:

Función	Tipo devuelto	Descripción
ArchivoExiste	Bool	Esta función permite verificar si el archivo que el usuario especificó en la lista de archivos existe o no
QuitarUno	Void	Esta es la función encargada de liberar el canal que está usando un usuario y que lo libera al dar doble clic en los canales que tiene reservado
SubirLimites	Void	Debido a que este es un control configurable, esta función es la encargada de subir al selector de banda ancha los límites que se usaran en la simulación
FijarSubMedio	Void	Esta función relaciona el medio que se va a usar en la simulación

Tabla 5.31 Funciones de la Clase Usuario001

CLASE Huffman: Clase encargada de realizar la compresión de la información y generar los valores para los cálculos estadísticos de la simulación.

Variables:

Variable	Tipo	Descripción
Control	String	Es una variable utilizada para verificar el correcto flujo al momento de crear el árbol
ArrayPesos	Long	Es un arreglo utilizado para controlar el peso de cada caracter y después utilizarlo en la construcción del árbol binario
Ramas	RamaNodo	Es un arreglo de tipo RamaNodo, este tipo contiene la información de la posición de cada uno de los caracteres que forman el árbol binario así como el nivel, el código y el peso de cada nodo
NumChr	Int	Es una variable utilizada para controlar el número de caracteres que contiene un archivo
PesoTotal	Long	En esta variable se almacena la información del peso total del archivo que se está codificando
NombreArchivo	String	Almacena la ubicación y el nombre del archivo a codificar

Tabla 5.32 Variables de la Clase Huffman

Funciones:

Función	Tipo devuelto	Descripción
Orderar	Void	Esta función permite ordenar el arreglo de Ramas del árbol según el peso (número de veces que se repite archivo)
TamañoArchivo	Long	Esta función devuelve el tamaño de archivo, previamente calculado por la función "CalcularArchivoPesos"
NumeroCaracteres	Int	Esta función devuelve el número de caracteres que contiene el archivo original
CalcularArchivoPesos	Long	Esta es la función encargada de calcular el peso del archivo original
CrearArbol	Int	Esta función utiliza los pesos de los caracteres que se encuentran en el arreglo "Ramas", crea un árbol binario según el código Huffman
SubirHojas	Int	Esta función permite tener todos los nodos que representan un caracter en las primeras posiciones del arreglo de ramas que forman el árbol binario
PesoTabla	Int	Esta función devuelve el peso de la tabla resultante al crear el árbol binario
PesoZip	int	Esta función devuelve el peso del archivo comprimido como resultado de aplicar Huffman

Tabla 5.33 Funciones de la Clase Huffman

CLASE LimitarCanales: Clase encargada de llevar el control del número de canales y de sus respectivos anchos de banda dependiendo del tipo de Trama con el que se está trabajando.

Variables:

Variable	Tipo	Descripción
vlLimite	Int	Esta variable almacena la cantidad de canales que da esta opción
vlNombre	String	Almacena el nombre que se mostrará en el ComboBox

Tabla 5.34 Variables de la Clase LimitarCanales

Funciones:

Función	Tipo devuelto	Descripción
LimitarCanales	Constructor	Esta es la función que se encarga de almacenar los datos en la variables respectivas
ToString	string	Es la sobrecarga de la función ToString, es la encargada de representar el nombre del medio

Tabla 5.35 Funciones de la Clase LimitarCanales

CLASE Program: Clase principal que instancia al formulario principal MDIParent

Funciones:

Función	Tipo devuelto	Descripción
Main	Void	Esta es la función inicial, es la encargada de crear el formulario principal y llamarlo

Tabla 5.36 Funciones de la Clase Program

CLASE RamaNodo: Clase encargada de crear el árbol binario para codificar la información comprimida.

Variables:

Variable	Tipo	Descripción
vPeso	Long	Representa el número de veces que se repite el caracter en el archivo a comprimir, este peso se lo utilizará luego para ordenar el árbol
vNivel	Int	Es el nivel de profundidad que tiene el caracter o la rama
vChr	Int	Es el número que representa el valor en ASCII del caracter
vSimbolo	String	Es el caracter que representa el valor
vCodigo	String	Representa el código del caracter en la codificación Huffman
vRamaI	RamaNodo	Es el objeto Rama Nodo que se encuentra a la Izquierda del caracter que reemplaza el nodo (Rama Izquierda)
vRamaD	RamaNodo	Es el objeto Rama Nodo que se encuentra a la Derecha del caracter que reemplaza el nodo (Rama Derecha)
vEs_Rama	Boolean	Esta variable indica si la rama se refiere a un caracter codificado a parte del código

Tabla 5.37 Variables de la Clase RamaNodo

Funciones:

Función	Tipo devuelto	Descripción
RamaNodo(RamaNodo RamaI, RamaNodo RamaD)	Constructor	Este constructor se utiliza para crear las ramas que no representan caracteres pero que ayudan en la codificación del árbol
RamaNodo(long Peso,int Chr)	Constructor	Este constructor se utiliza para crear las ramas que representan a caracteres y que están siendo codificados
GetPeso	Long	Devuelve el valor actual del peso de la rama
GetChr	Int	Devuelve un valor numérico que es la representación del caracter en la tabla ASCII
GetCodigo	String	Devuelve la cadena de “0” y “1” que representan el código del caracter codificado
GetSimbolo	String	Devuelve el caracter codificado
GetNivel	Int	Devuelve el nivel en que se encuentra actualmente la rama
GetEsRama	Bool	Esta es una función que utilizamos para diferenciar las ramas del árbol binario de sus hojas. Las ramas son nodos que no representan caracteres y las hojas son nodos que si representan caracteres
PlusNilvel	Void	Función encargada de incrementar el contador de nivel de profundidad del árbol
SetCodigo	Void	Esta función permite grabar un código enviado en la variable local vl_codigo
BajarNivel	Void	Esta es la función encargada de mover las ramas a la izquierda o a la derecha al momento de la creación del árbol binario, una vez que la mueve baja todo el sub-árbol un nivel.

Tabla 5.38 Funciones de la clase RamaNodo

Reportes gráficos

ReportGraficoTiempos: Este reporte muestra una grafica comparativa de los tiempos en segundos que demora todo el proceso de envío de los archivos seleccionados.

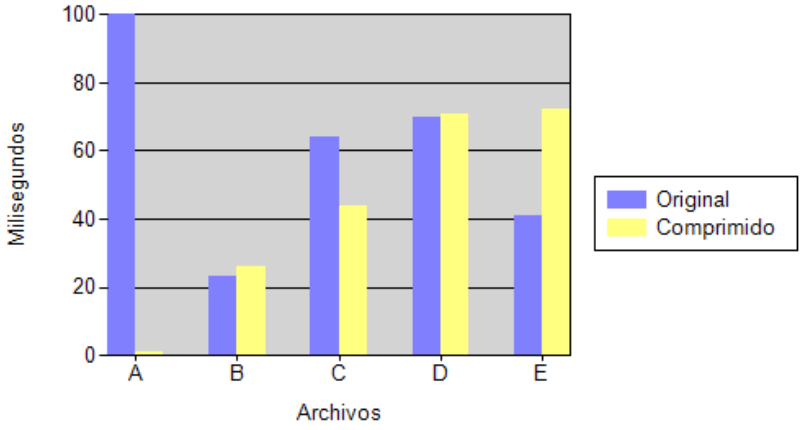


Fig. 5. 29 Reporte gráfico de las estadísticas de compresión en función del tiempo

Rpt_est_Calculos: Este es el gráfico que muestra los resultados de toda la simulación del envío de los archivos, además muestra un resumen de la sumatoria total de todos los archivos por usuario.

ombre arhivo	Peso (bytes)	Comprimido (bytes)	Tabla (bytes)	Medio de Transmicion	Distancia(km)	Ancho Banda (kbps)	Ttx (ms)	Tprop (ms)	Tproc (ms)	T(Segundo)	T(Minuutos
=Fields!Est_nom	=Fields!Est_peso.Val	=Fields!Est_Comprimido.\	=Fields!Est_Tabla.\	=Fields!Est_Medio.Value	=Fields!Est_Distancia.	=Fields!Est_AnchoBanda.Va	=Round((Fields!Est_pes	=Round(Fields!Est_Distani	=Round((((16*(((Fields!Es	=Round((((Fields	=Round((((Fie
TOTALES:	=Sum(Fields!	=Sum(Fields!	=Sum(Fields!							=Round(Sum	=Round(Sum

Fig. 5. 30 Resumen estadístico de los cálculos de los tiempos de transmisión

Rpt_Grafico: Gráfico que permite comparar los tamaños de los archivos originales con su semejante codificado por el método de Huffman.

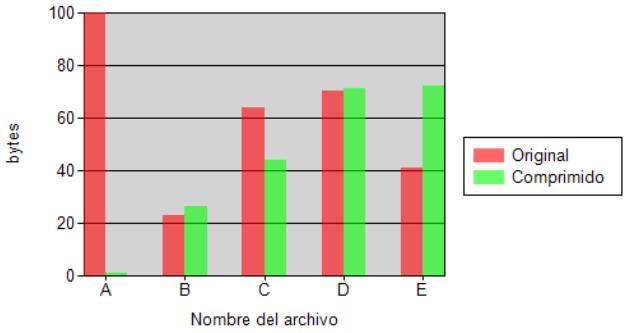


Fig. 5. 31 Reporte gráfico de las estadísticas de compresión en función del tamaño de archivo

Base de Datos

DBEstadisticas: Almacena los resultados de la simulación para crear las gráficas de los mismos, para ello se utiliza un archivo de base de datos de Access.

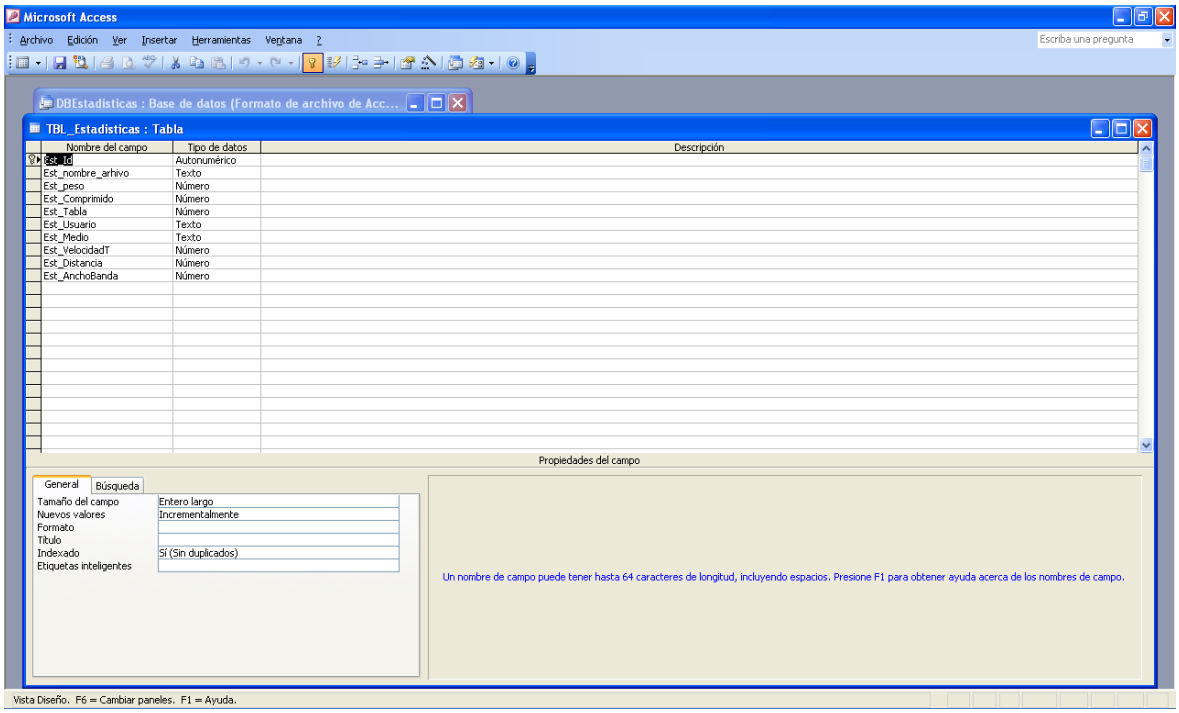


Fig. 5. 32 Vista de la base de datos donde se almacenan las variables que intervienen en los cálculos de compresión.

CAMPOS DE LA TABLA DE LA BASE DE DATOS

Campo	Tipo	Descripción
Est_Id	Autonumérico	Este es identificador único utilizado por la base de datos como clave principal
Est_nombre_archivo	Texto	Almacena el nombre del archivo a codificar
Est_peso	Número	Almacena el peso original del archivo
Est_comprimido	Número	Almacena el peso del archivo resultante de aplicar la compresión de Huffman
Est_tabla	Número	Almacena el peso de la tabla resultante de la compresión, esta tabla contiene los caracteres y los códigos asociados para la decodificación
Est_Usuario	Texto	Almacena el id del usuario, esto permite agrupar los archivos según el usuario que los envía
Est_Medio	Texto	Almacena el nombre del medio que se utiliza para la transmisión de datos
Est_VelocidadT	Número	Almacena la velocidad de propagación según el medio de transmisión
Est_Distancia	Número	Almacena la distancia a la que se encuentran los puntos de transmisión y recepción
Est_AnchoBanda	Número	Almacena el ancho de banda que se asignó para la transmisión del archivo

Tabla 5.39 Campos que conforman la base de datos

5.4 FASE DE PRUEBAS Y CORRECCIONES

Dentro de la fase de pruebas y correcciones se toma en cuenta los siguientes aspectos:

- Tratamiento de objetos para definir el tipo de tecnología y usuarios en un mismo formulario. Con esto se evita tener varios formularios para cada tecnología de transmisión.
- Tratamiento de objetos para definir el número de canales de acuerdo a la trama seleccionada, con esto se evita tener varios formularios para cada tecnología de transmisión.

Estas son las correcciones que se ha definido para que la aplicación sea modular.

5.5 SIMULACIÓN EN LA RED

Esta aplicación no se ajusta para una simulación en una red físicamente estructurada, al contrario se ajusta a una simulación local, por cuanto la aplicación es orientada más a cálculos estadísticos en cuestión del proceso de transmisión de datos.

Se trata de una aplicación para comprender como se lleva a cabo el proceso de la transmisión de datos por enlaces SDH y PDH, mediante esto, el usuario puede conocer los principios básicos de estas dos tecnologías de transporte y comprender las ventajas y desventajas de introducir algoritmos de compresión de datos en la transmisión.

5.6 MANUAL DE USUARIO

Para que la instalación del Software Aplicación del Algoritmo de Compresión basado en Huffman en su máquina sea satisfactoria es necesario que tome en cuenta los siguientes puntos.

¿Qué es el Sistema de compresión Huffman?

El Sistema de Compresión basado en el Algoritmo de Huffman es un algoritmo que busca comprimir archivos (información) formando un árbol binario tomando en cuenta la frecuencia de repetición de la información. Es decir a la información que más se repite se le asigna un código mucho más corto que a la información que menos se repite, de esta manera se codifica la información de una manera eficiente.

Especificaciones Técnicas para requerimientos de Hardware y Software

Los requerimientos mínimos de hardware y software son:

• Procesador Pentium III (Compatible) o superior
• Mínimo 200 MB libres en disco duro
• Instalar el utilitario .Net Framework3.5
• Instalar Flash Player

Compatibilidad de Sistemas Operativos

Esta aplicación es compatible con los siguientes sistemas operativos:

• Windows XP
• Windows Vista

INSTALACIÓN DE HUFFMAN

Para iniciar la instalación de la Aplicación Huffman se debe hacer doble click sobre el ícono de Setup dentro de la carpeta de instalación.

Fig. 5. 33 Icono de instalación

El proceso de instalación se iniciará y mostrará la ventana de instalación. Seleccionar la opción Instalar.

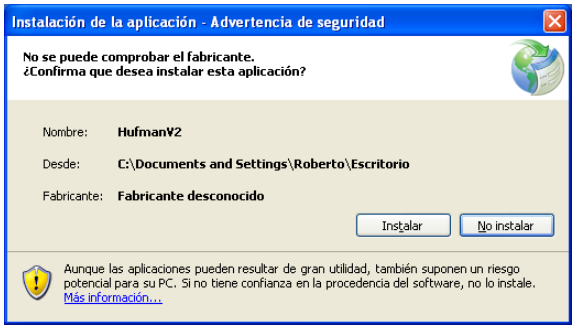


Fig. 5. 34 Advertencia de seguridad

Se carga la aplicación denominada HuffmanV2 en el equipo.

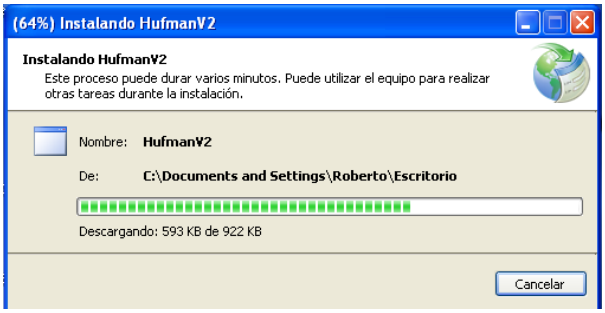


Fig. 5. 35 Proceso de instalación

Al finalizar la instalación se mostrará la pantalla principal de la aplicación.

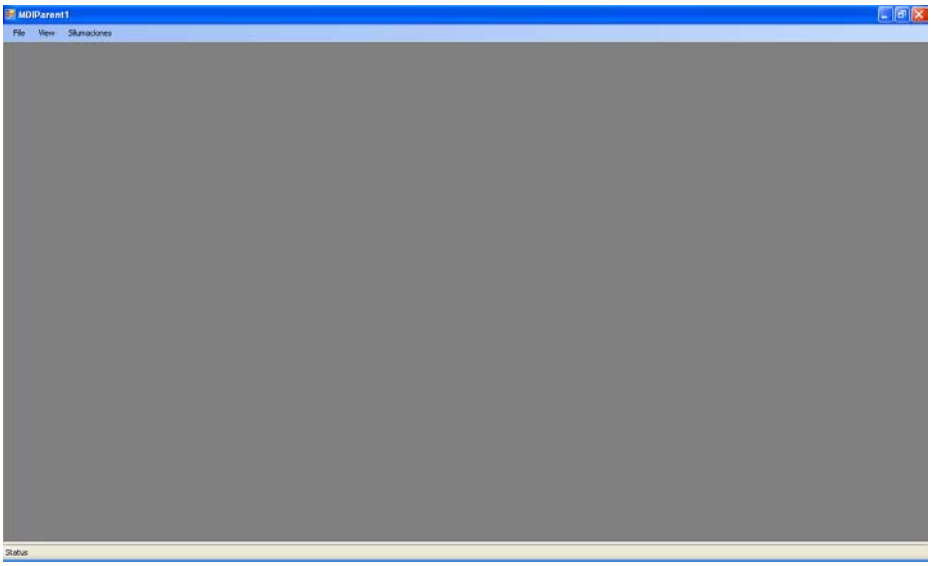


Fig. 5. 36 Pantalla inicial

En el menú de programas se visualiza la aplicación HuffmanV2

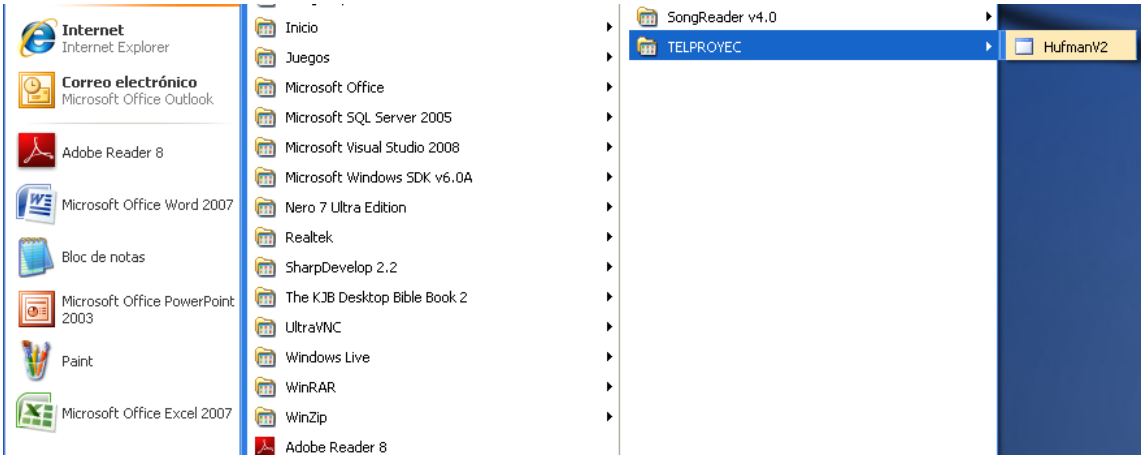


Fig. 5. 37 Verificando la instalación de la aplicación

Para desinstalar la aplicación se debe hacerlo mediante el panel de control opción Agregar o Quitar Programas

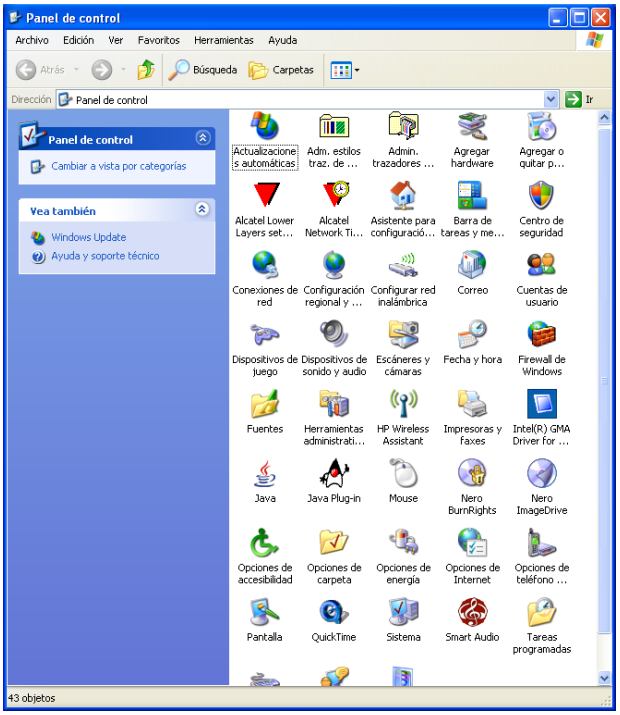


Fig. 5. 38 Proceso de desinstalación de la aplicación

Seleccionar HuffmanV2 y Click en Cambiar o quitar

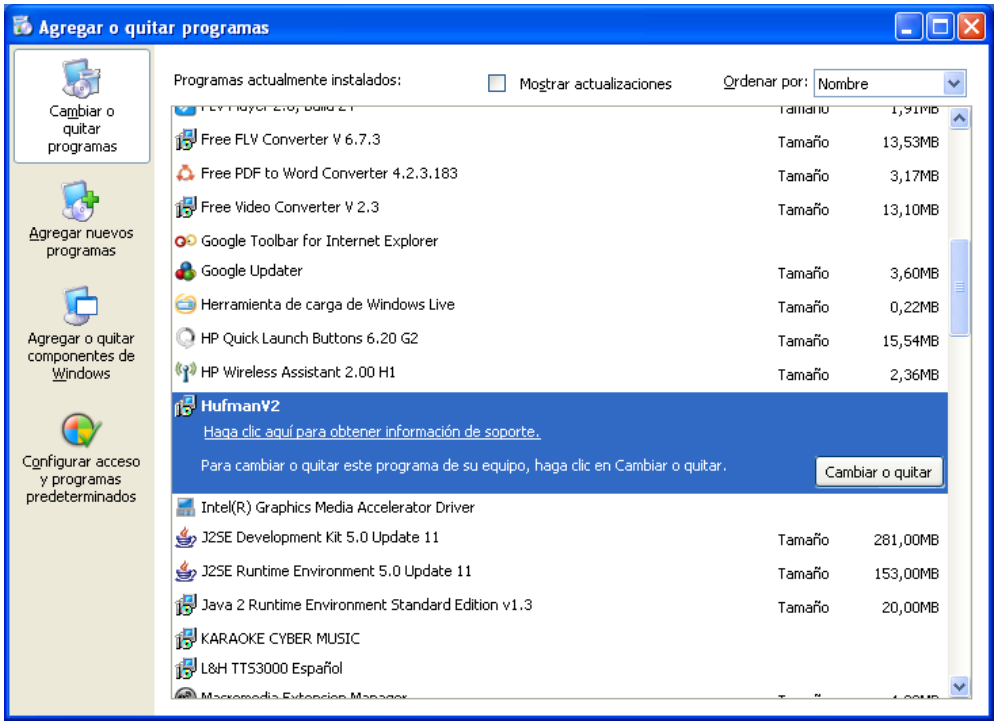


Fig. 5. 39 Quitando el programa Huffman

Para desinstalar la Aplicación Click en Aceptar

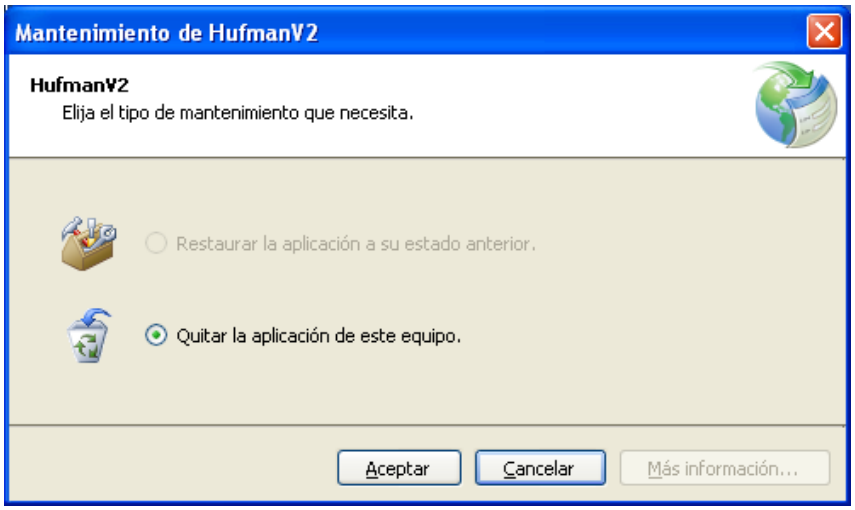


Fig. 5. 40 Proceso de desinstalación de la aplicación

FUNCIONALIDAD DE LA APLICACIÓN.

Formulario Principal

El formulario principal en su barra de menús contiene las siguientes opciones:

Menú File.

Opción Exit:

Cierra el Formulario Principal.

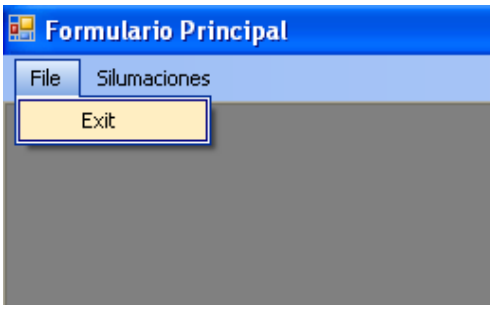


Fig. 5. 41 Opción Salir de la aplicación

Menú Simulaciones.

Opción Archivo en E1:

Muestra la asignación de la información en una trama E1.

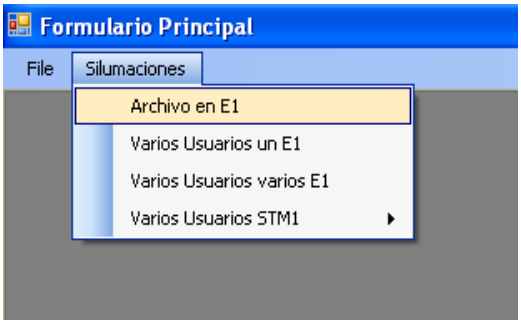


Fig. 5. 42 Opción Archivo en E1

Al elegir esta opción se visualiza una pantalla donde se puede ver cómo se asigna la información en los diferentes canales de una trama E1. Además se visualiza el tamaño del archivo seleccionado.

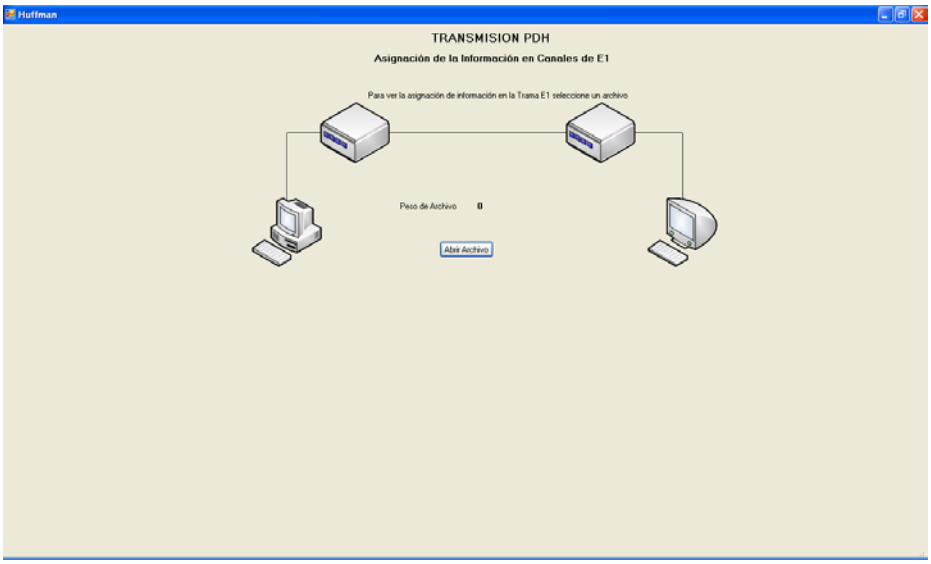


Fig. 5. 43 Interfaz de asignación de información a canales de una trama E1

Para ello se requiere seleccionar un archivo. Click en el botón Abrir Archivo.

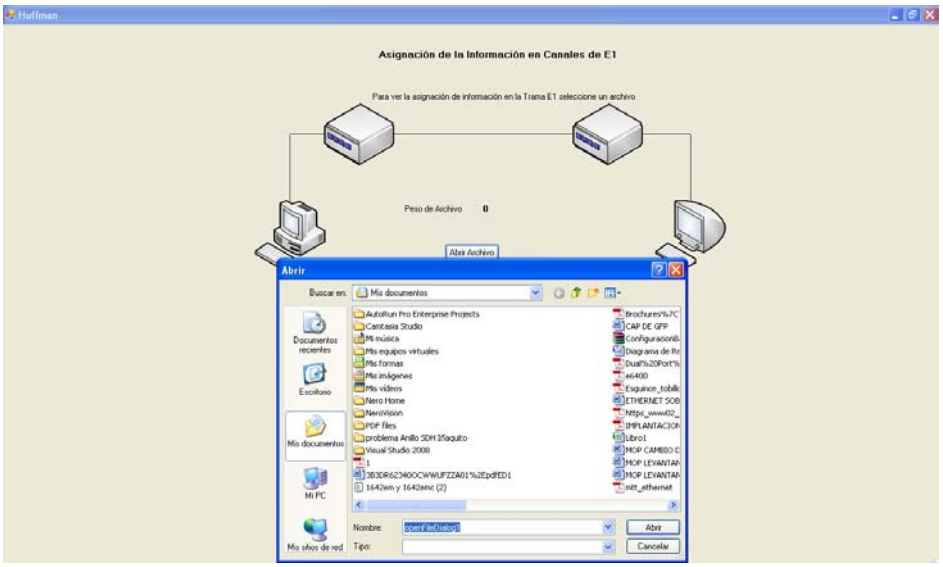


Fig. 5. 44 Selección del archivo de prueba

Seleccionar un archivo para visualizar la asignación de la información en los canales de la Trama E1 de la manera como muestra la figura.

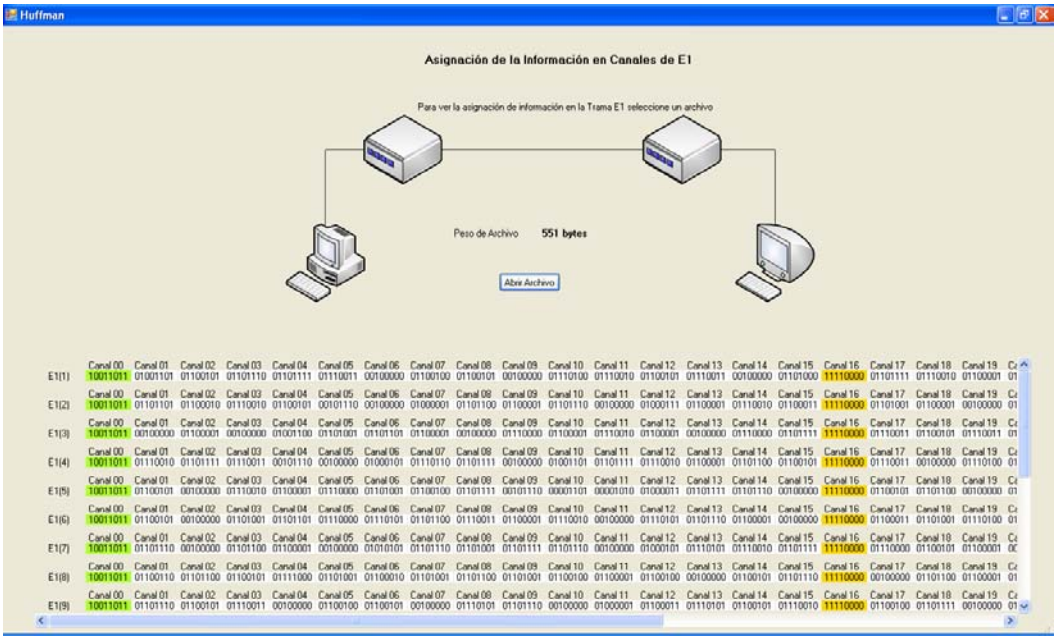


Fig. 5. 45 Asignación de la información en los canales de la trama E1

Opción Varios Usuarios un E1:

Muestra un formulario donde se simula el envío de información codificada de varios usuarios sobre una única trama E1

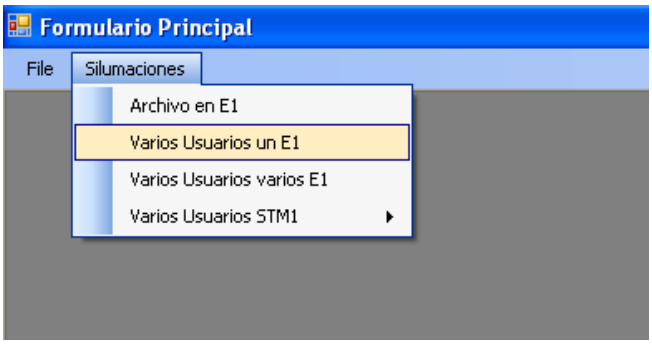


Fig. 5. 46 Opción varios Usuarios interactuando con un E1

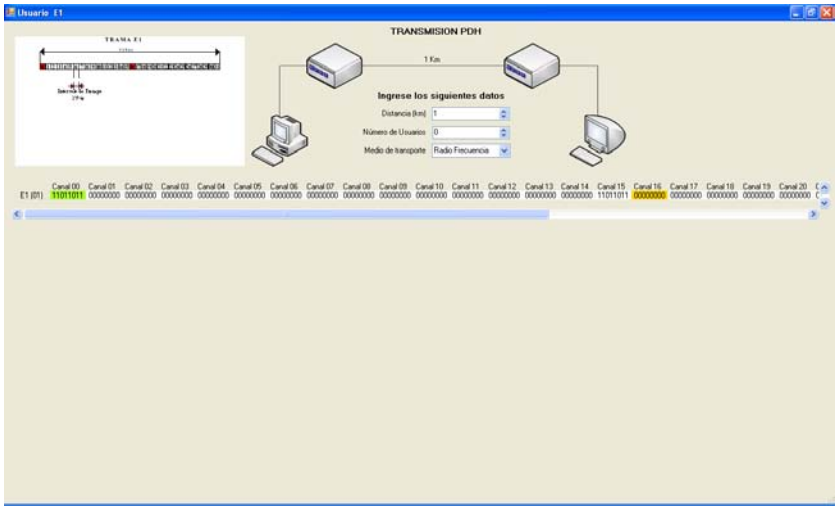


Fig. 5. 47 Interfaz de usuario para la selección de los parámetros de transmisión

El formulario consta de una sección para el ingreso de las variables que intervienen en la transmisión como: distancia, número de usuarios y medio de transporte.

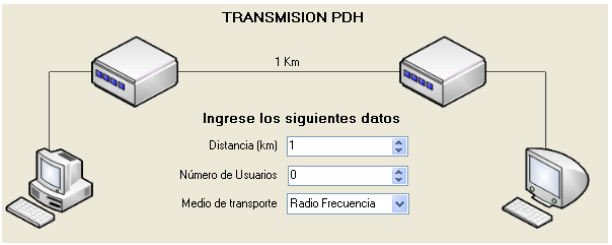


Fig. 5. 48 Ingreso de las variables que intervienen en la transmisión

Adicional de una sección donde se visualiza una animación ilustrativa que muestra como se forma una Trama E1.

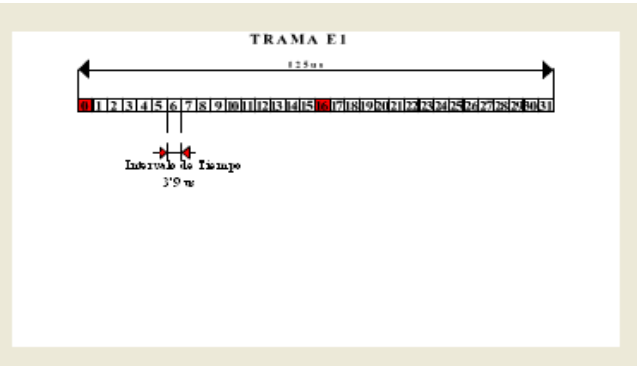


Fig. 5. 49 Animación que ejemplifica la formación de una trama E1

También se visualiza la Trama E1 y sus canales a los cuales se les asignarán un usuario dependiendo del ancho de banda seleccionado.

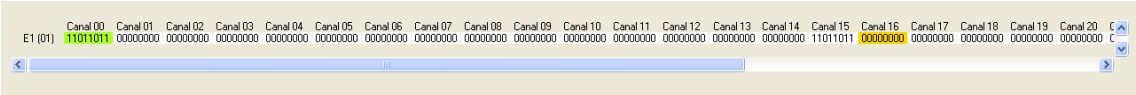


Fig. 5. 50 Formación de una trama E1

Adicional una sección para la interfaz del Usuario que consta de dos pestañas Transmisión y Huffman.

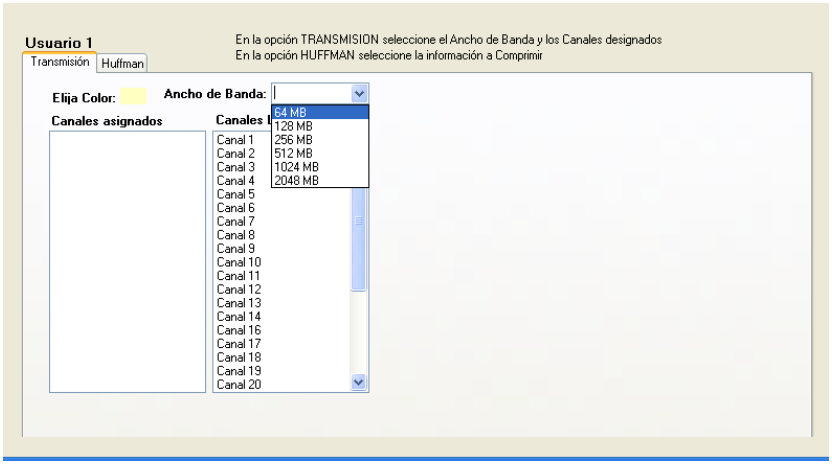


Fig. 5. 51 Selección de los canales del Ancho de Banda

Para iniciar se debe seleccionar las variables que intervienen en la Transmisión:

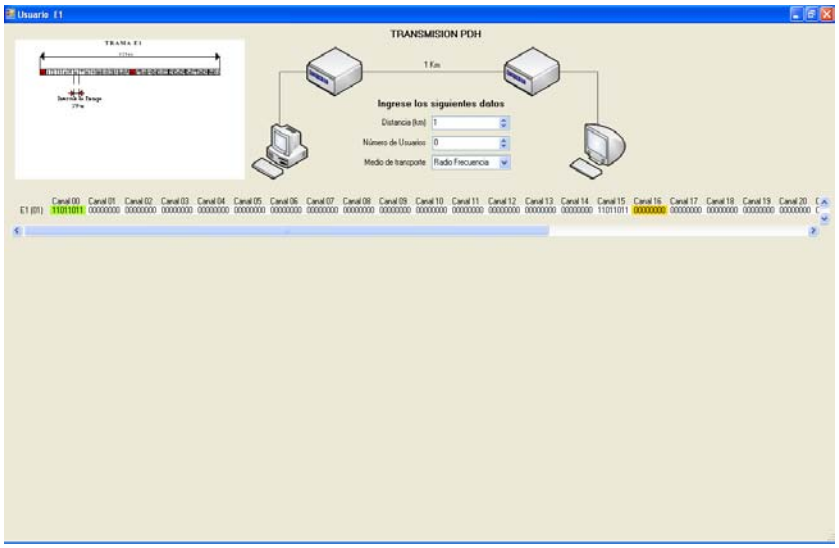


Fig. 5. 52 Vista general

Al seleccionar el número de usuarios se visualiza la interfaz del usuario misma que consta de dos pestañas (Transmisión y Huffman).

En la opción Transmisión se elige el Ancho de Banda y los canales de la Trama E1 que se asignará al usuario.

Para seleccionar el Ancho de Banda se selecciona de la lista. Para elegir el canal se debe dar doble Click sobre el listado de Canales Disponibles, de este modo se visualizará el canal seleccionado pintado de diferente color para cada usuario.

El listado de Canales Libres se actualiza de inmediato, con esto se controla que un usuario no elija un canal ocupado.

En la opción Huffman se debe elegir la información que se comprimirá y simulará el envío. Para ello Click en el botón Agregar y elegir los archivos a codificar.

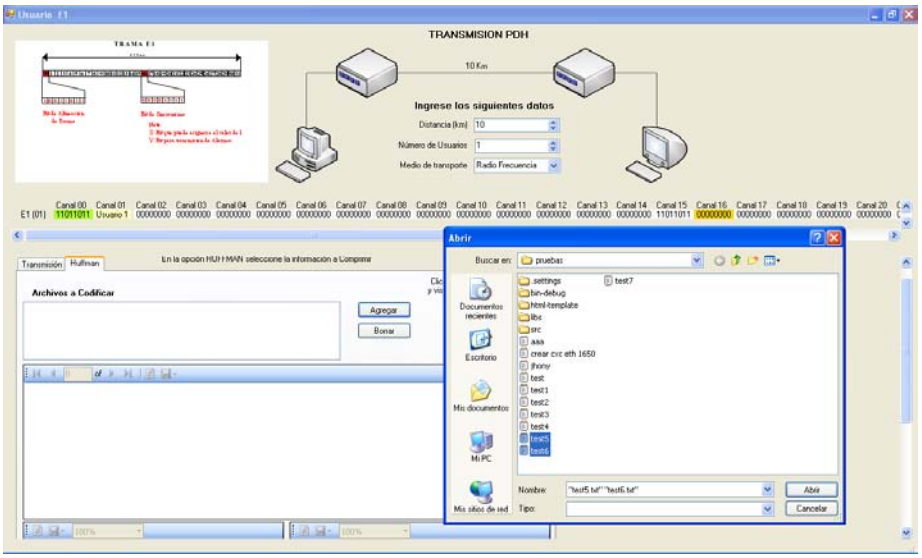


Fig. 5. 53 Elección de los archivos que intervienen en el proceso de la compresión

La información seleccionada se lista en el área de Archivos a Codificar, en esta área se puede Agregar o Borrar archivos a comprimir. Para borrar se debe seleccionar el archivo de la lista y luego pulsar el botón Borrar.

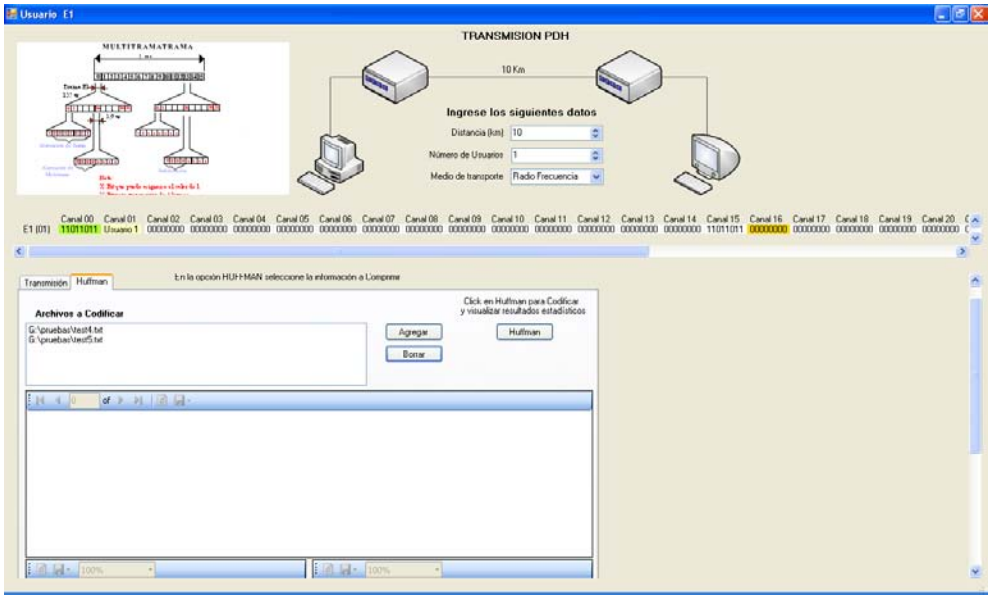


Fig. 5. 54 Vista de la interfaz de usuario, elección de archivos a comprimir

Una vez listada toda la información a codificar se procede a comprimirla. Para ello se debe hacer Click sobre el botón Huffman. Al hacer esta acción se visualizan las estadísticas de compresión y simulación de transmisión de los archivos seleccionados. Las estadísticas se muestran en forma de tabla y en forma gráfica.

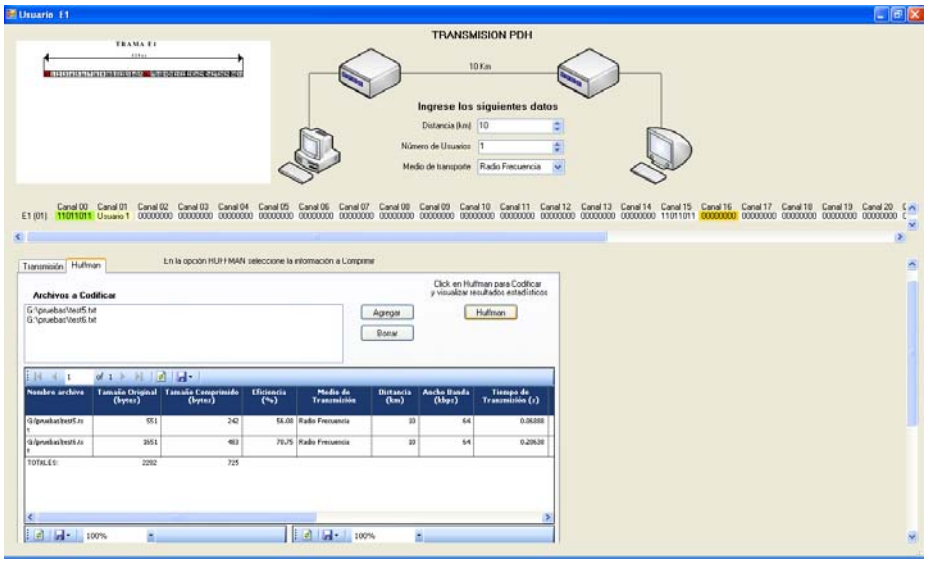


Fig. 5. 55 Estadísticas de compresión

En el área de las estadísticas ya sea de forma de tabla y gráfico se tiene la posibilidad de exportar los datos a formato Excel y PDF.

Exportar los datos a Excel

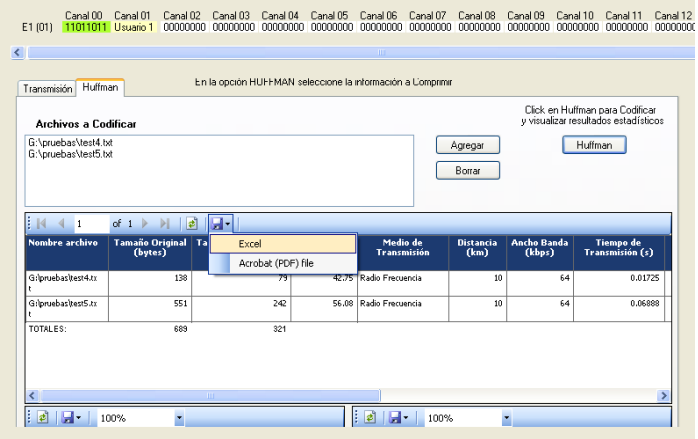


Fig. 5. 56 Se exporta a Excel las estadísticas obtenidas luego de la compresión

Exportar los datos a PDF

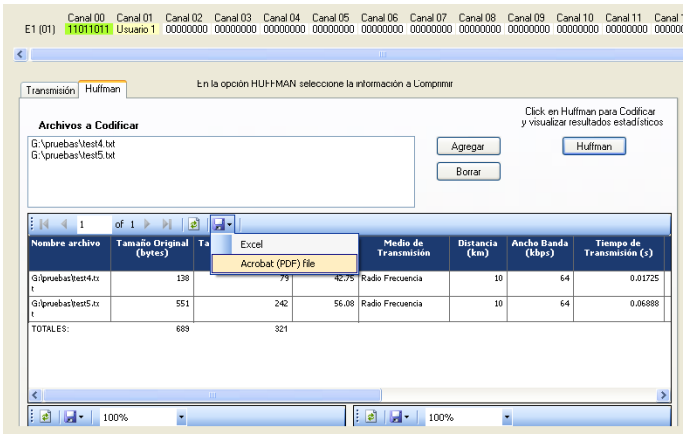


Fig. 5. 57 Se exporta a PDF las estadísticas obtenidas luego de la compresión

Resultados exportados a formato EXCEL

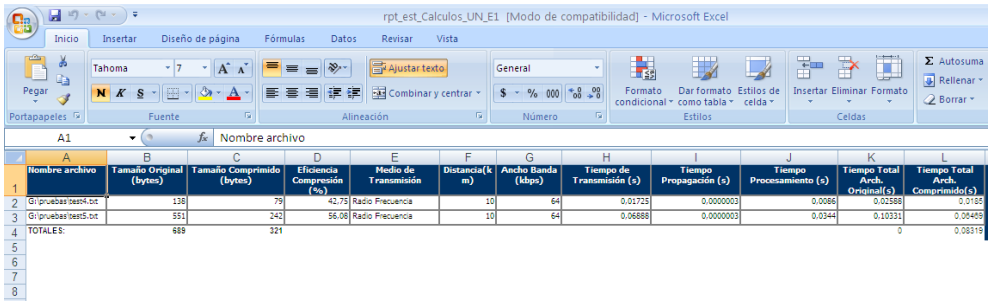


Fig. 5. 58 Resultados exportados a formato Excel

Resultados exportados a formato PDF

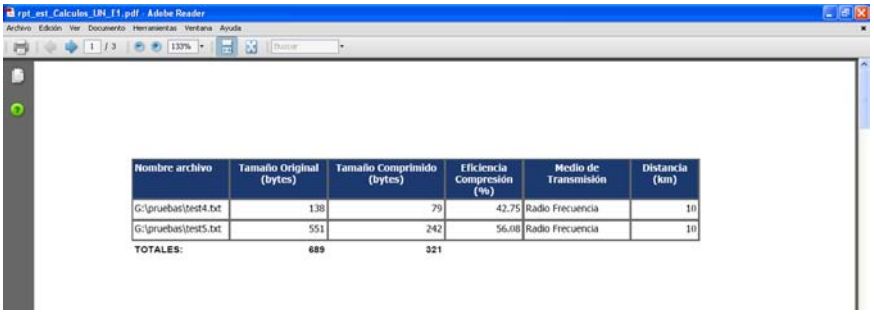


Fig. 5. 59 Resultados exportados a PDF

En todos los reportes se da la posibilidad de refrescar los datos.

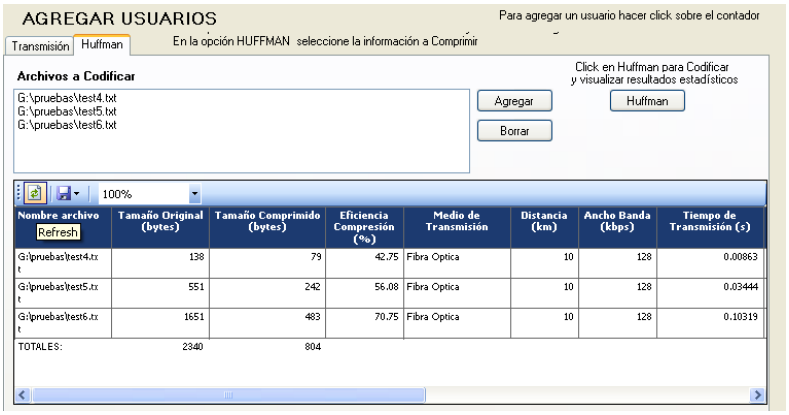


Fig. 5. 60 Refrescar los datos de la tabla

En todos los reportes se da la posibilidad de ampliar o disminuir la vista de acuerdo a las necesidades del cliente.

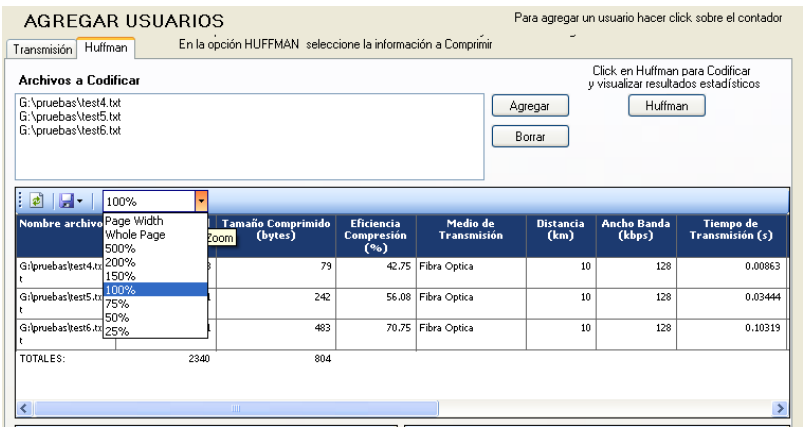


Fig. 5. 61 Ajustar el tamaño de visualización

Opción Varios Usuarios varios E1:

Muestra un formulario donde se simula el envío de información codificada de varios usuarios sobre varias tramas E1. Esta simulación permite aumentar Tramas E1 a medida que aumentan los usuarios de una red.

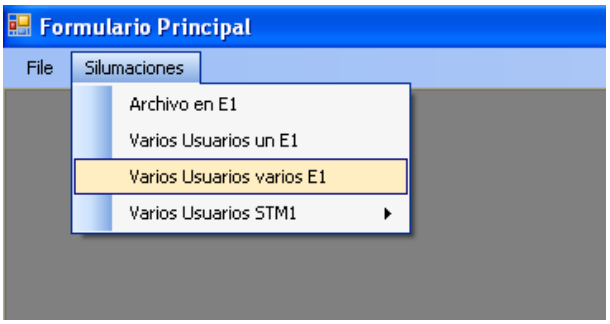


Fig. 5. 62 Opción varios Usuarios interactuando con varios E1

Al seleccionar esta opción se despliega una ventana donde se tendrá que ingresar las variables que intervienen en la transmisión como: distancia, número de usuarios y medio de transporte.

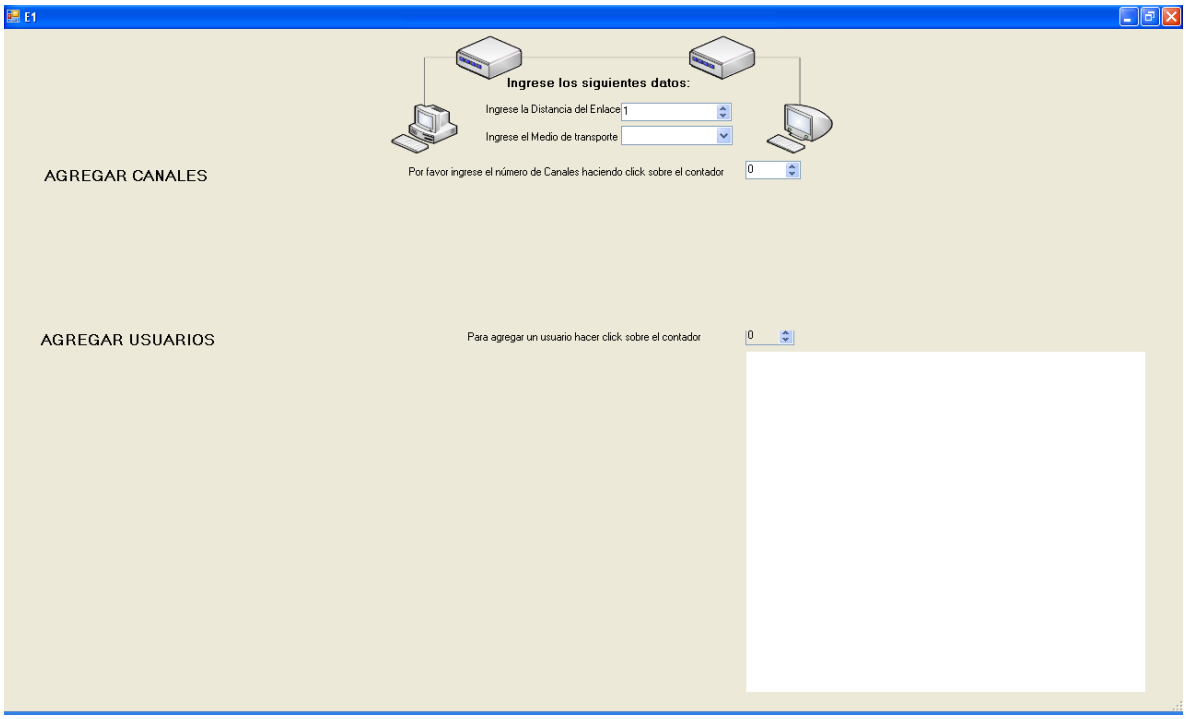


Fig. 5. 63 Interfaz de usuario para la selección de los parámetros de transmisión

El formulario consta de una sección para el ingreso de las variables que intervienen en la transmisión como: distancia y medio de transporte.

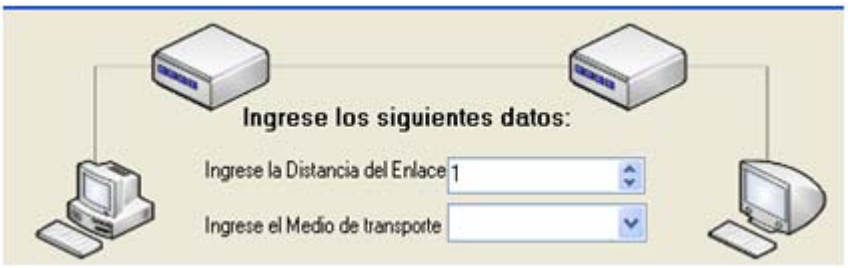


Fig. 5. 64 Ingreso de las variables que intervienen en la transmisión

El formulario tiene un espacio definido donde se visualizan las Tramas E1 y sus canales a los cuales se les asignarán un usuario dependiendo del ancho de banda seleccionado.

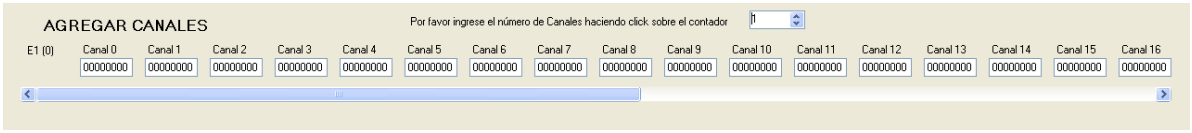


Fig. 5. 65 Formación de una trama E1 dinámica

También dispone de un área para la interfaz de usuarios y una sección donde se visualiza una animación ilustrativa que muestra como se forma una Trama E1.

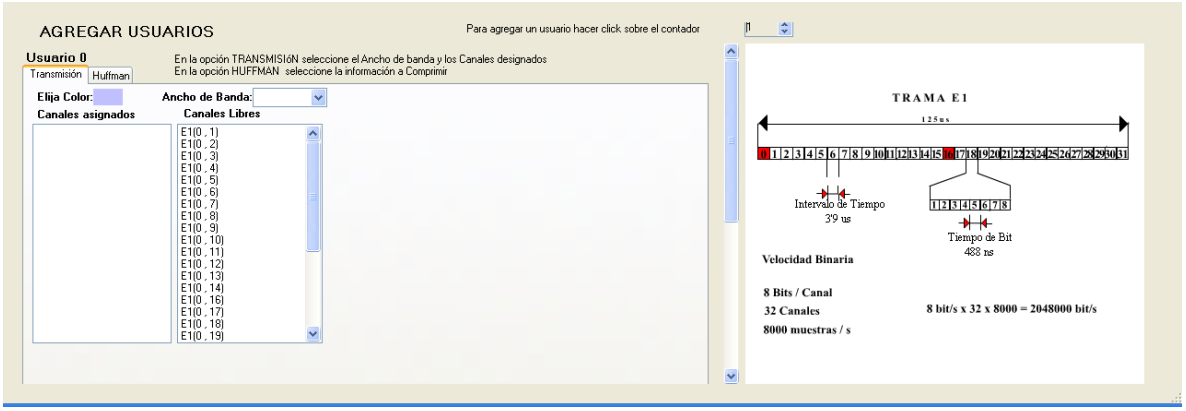


Fig. 5. 66 Selección de los canales del Ancho de Banda

La interfaz de usuario dispone de dos pestañas Transmisión y Huffman.

La opción Transmisión permite elegir el Ancho de Banda y los canales de los usuarios.

La opción Huffman permite elegir la información que se codificará y simulará su envío, además se puede ver los resultados estadísticos del proceso de transmisión.

Ingreso de las variable que intervienen en la transmisión.

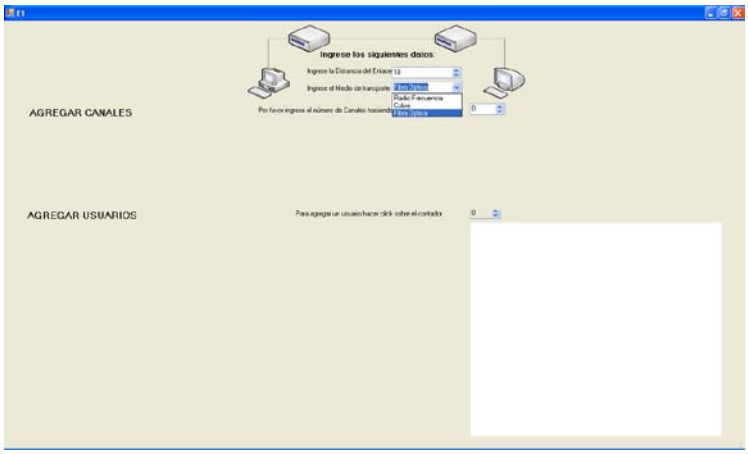


Fig. 5. 67 Selección de las variables de transmisión

Al elegir el número de Tramas E1 se carga el control que permite agregar Tramas E1 a nuestro formulario.

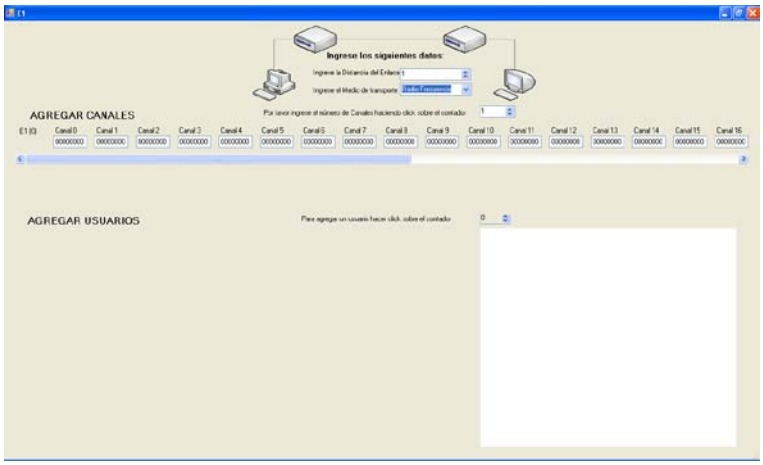


Fig. 5. 68 Agregar usuarios

Al incrementar el número de usuarios se carga la animación donde se visualiza la formación de la Trama E1 y se habilita las opciones de Transmisión y Huffman.

En la opción Transmisión se debe elegir el Ancho de Banda y el canal de la Trama E1 designado para el usuario.

Para elegir el Ancho de Banda se selecciona de la lista. Para elegir el canal se debe dar doble Click sobre el listado de Canales Disponibles, de este modo se visualizará el canal seleccionado pintado de diferente color para cada usuario.

El listado de Canales Libres se actualiza de inmediato, con esto se controla que un usuario no elija un canal ocupado.

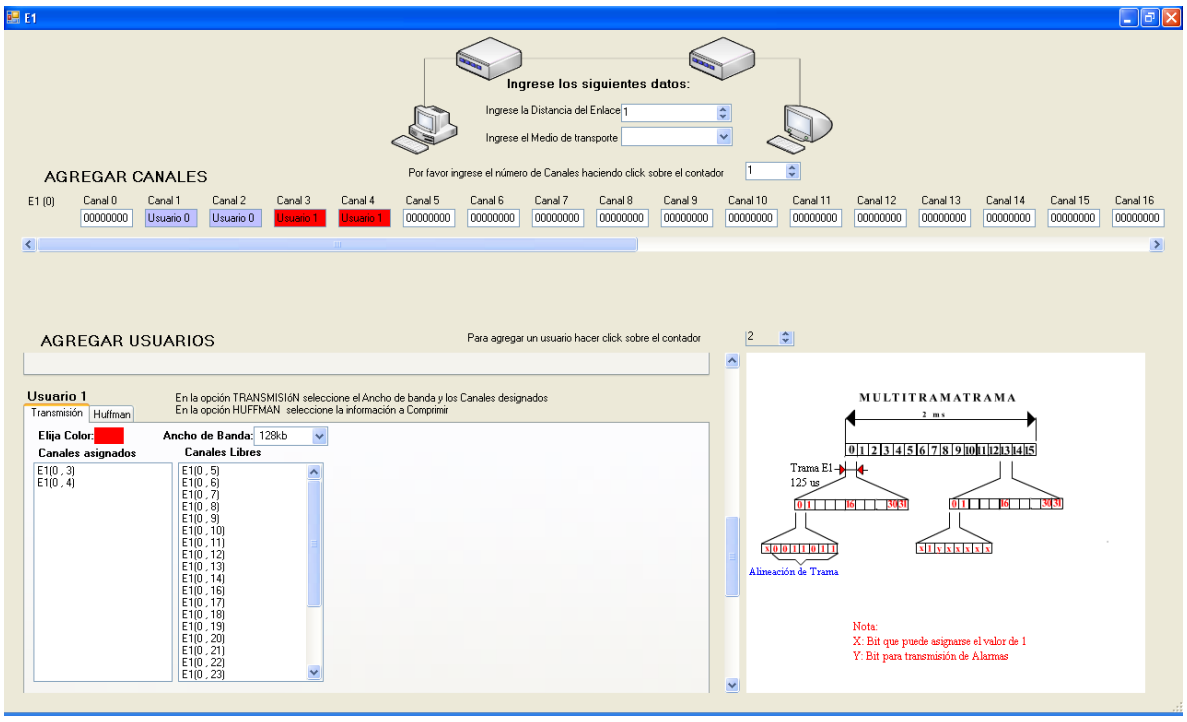


Fig. 5. 69 Selección de los canales del Ancho de Banda

En la opción Huffman se debe elegir la información que se comprimirá y simulará el envío. Para ello Click en el botón Agregar y elegir los archivos a codificar.

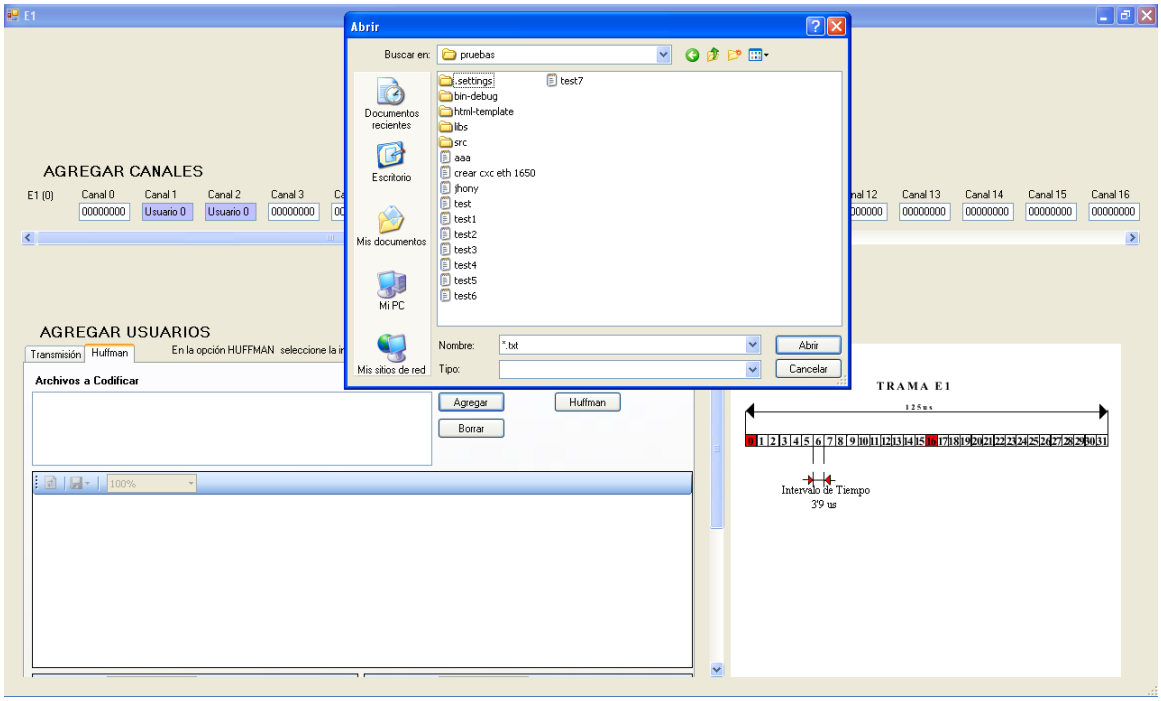


Fig. 5. 70 Elección de los archivos que intervienen en el proceso de la compresión

La información seleccionada se lista en el área de Archivos a Codificar, en esta área se puede Agregar o Borrar archivos a comprimir. Para borrar se debe seleccionar el archivo de la lista y luego pulsar el botón Borrar.

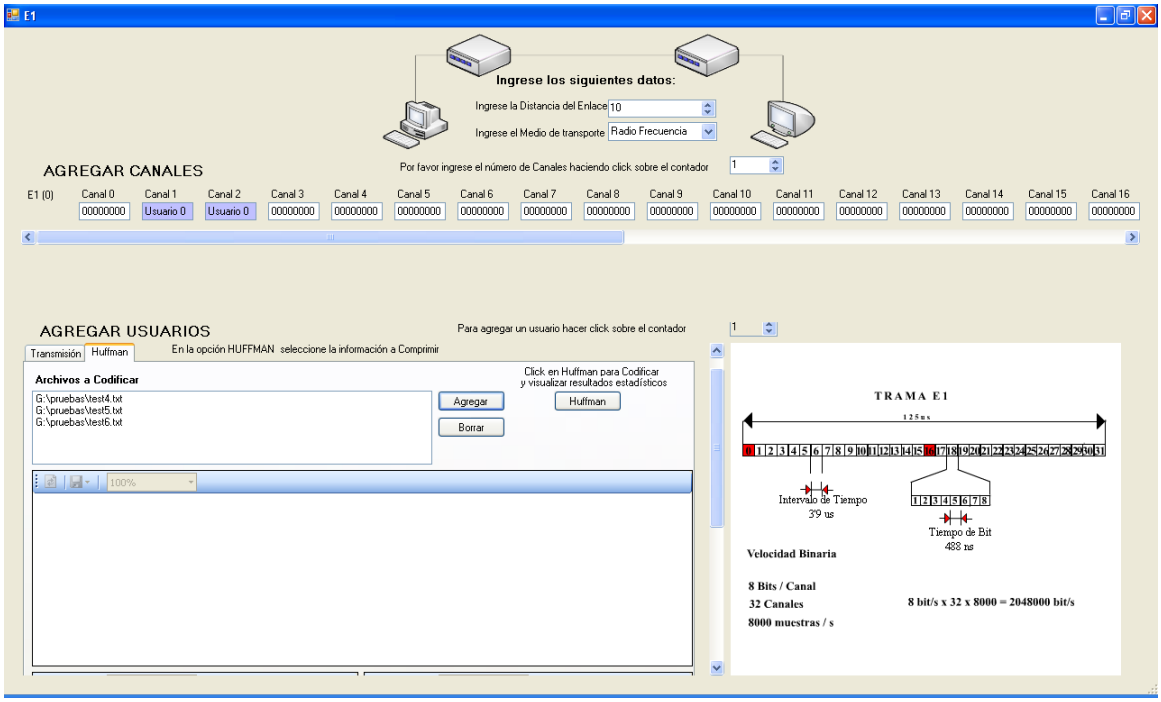


Fig. 5. 71 Vista de la interfaz de usuario, elección de archivos a comprimir

Una vez listada toda la información a codificar se procede a comprimirla. Para ello se debe hacer Click sobre el botón Huffman.

Al hacer esta acción se visualizan las estadísticas de compresión y simulación de transmisión de los archivos seleccionados. Las estadísticas se muestran en forma de tabla y en forma gráfica.

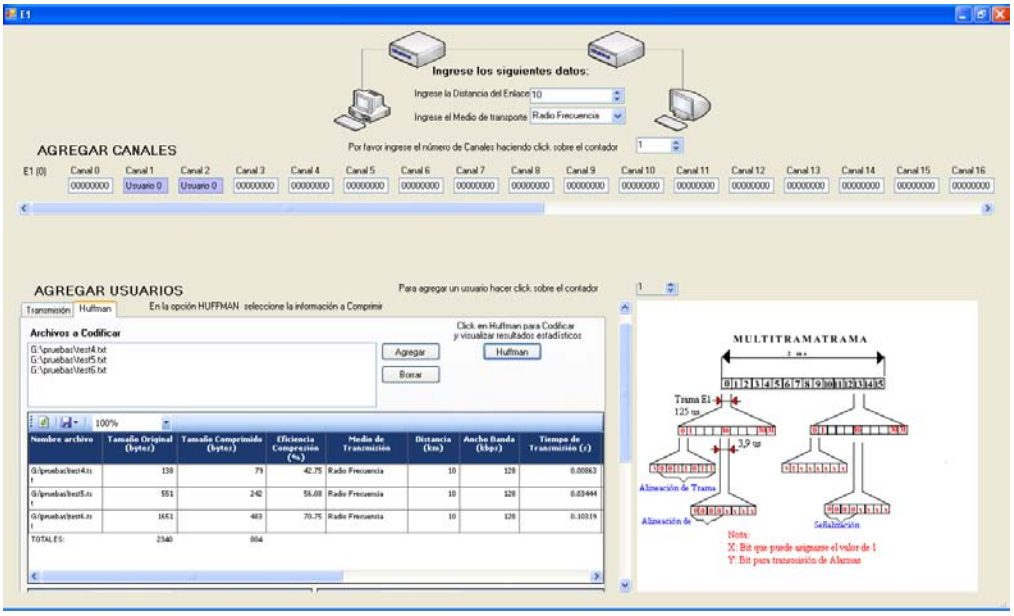


Fig. 5. 72 Estadísticas de compresión

En el área de las estadísticas ya sea de forma de tabla y gráfico se tiene la posibilidad de exportar los datos a formato Excel y PDF.

Exportar los datos a Excel

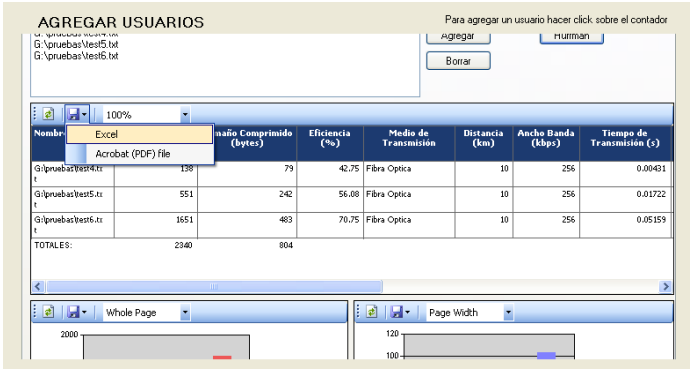


Fig. 5. 73 Se exporta a Excel las estadísticas obtenidas luego de la compresión

Exportar los datos a PDF

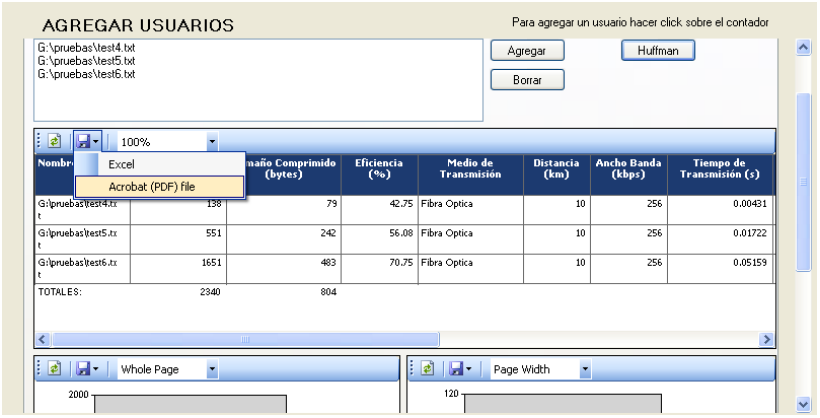


Fig. 5. 74 Se exporta a PDF las estadísticas obtenidas luego de la compresión

Resultados exportados a formato PDF

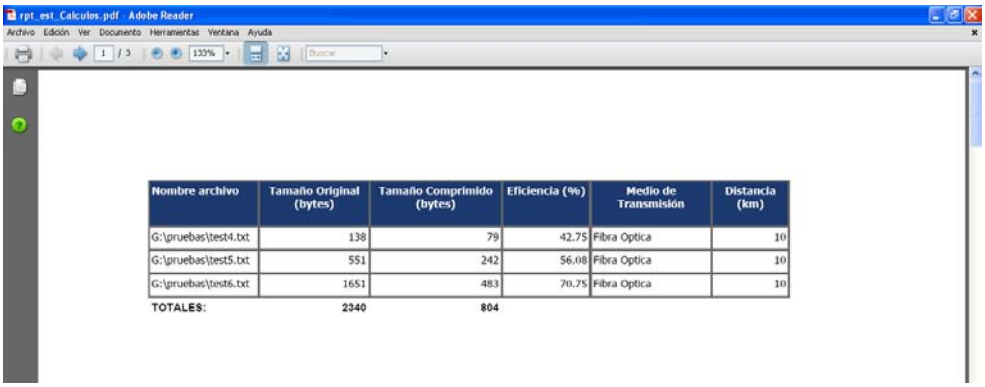


Fig. 5. 75 Resultados exportados a PDF

Resultados exportados a formato EXCEL

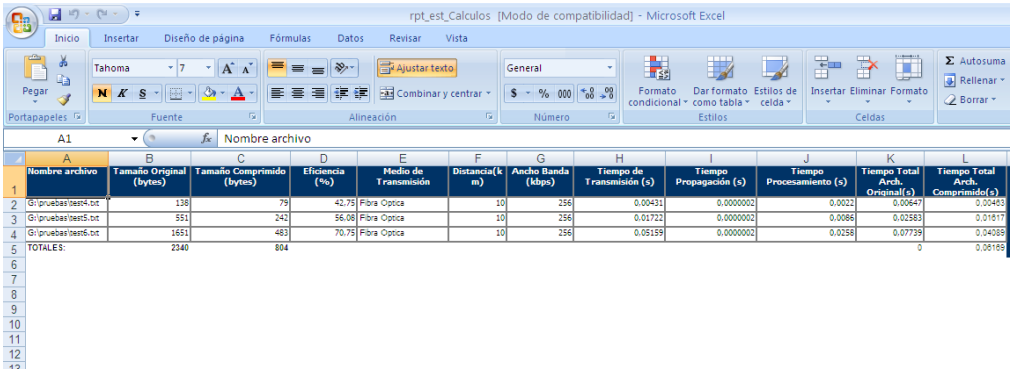


Fig. 5. 76 Resultados exportados a formato Excel

En todos los reportes se da la posibilidad de refrescar los datos.

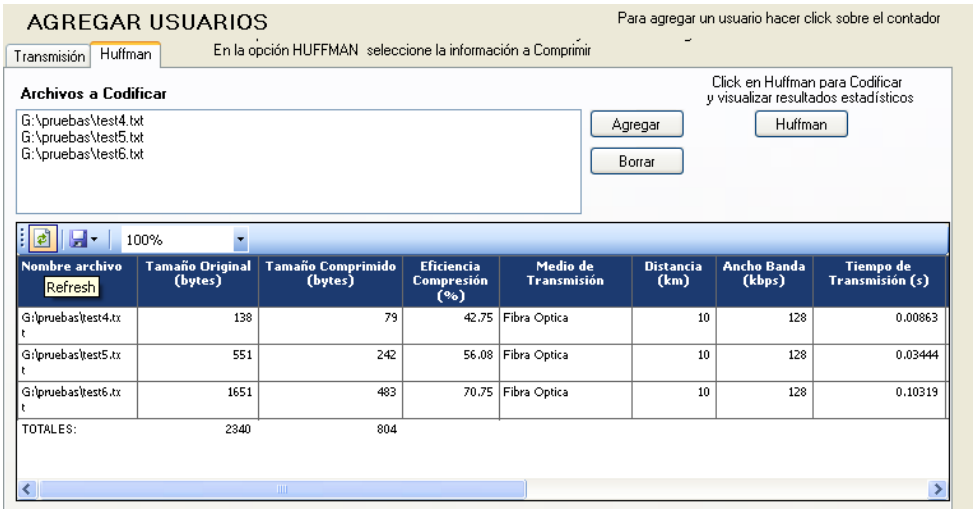


Fig. 5. 77 Refrescar los datos de la tabla

En todos los reportes se da la posibilidad de ampliar o disminuir la vista de acuerdo a las necesidades del cliente.

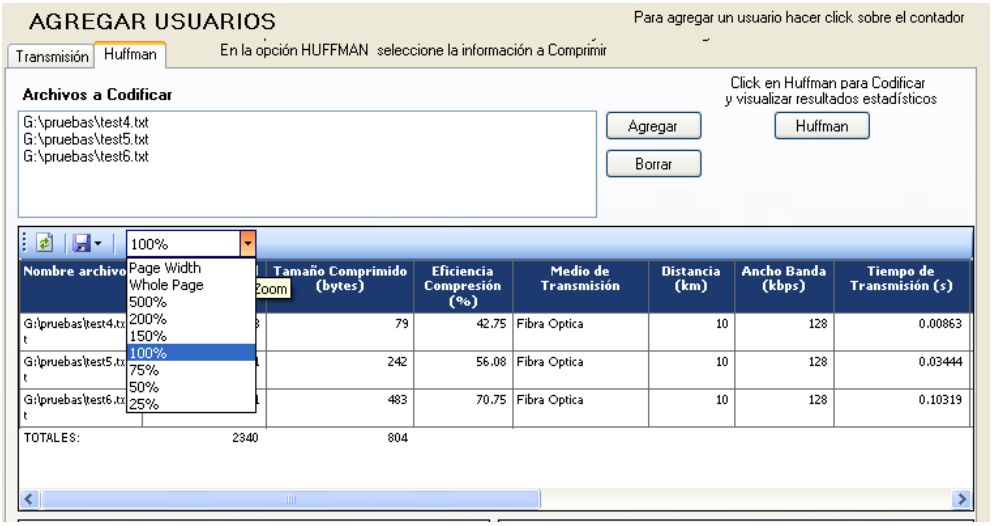


Fig. 5. 78 Ajustar el tamaño de visualización

Opción Varios Usuarios STM1 -> TU-12:

Muestra un formulario donde se simula el envío de información codificada de varios usuarios sobre varias tramas STM-1 en Unidades Tributarias TU-12. Esta simulación permite aumentar Tramas STM-1 a medida que aumentan los usuarios de una red.

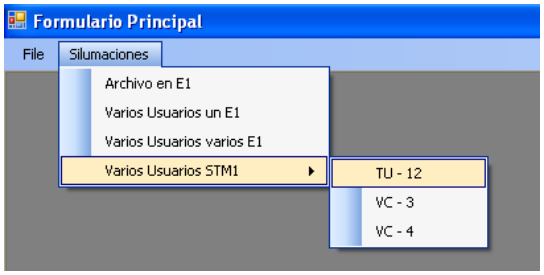


Fig. 5. 79 Opción varios Usuarios interactuando con varios E1

Al seleccionar esta opción se despliega una ventana donde se tendrá que ingresar las variables que intervienen en la transmisión como: distancia, medio de transporte, número de usuarios y número de Tramas STM-1.

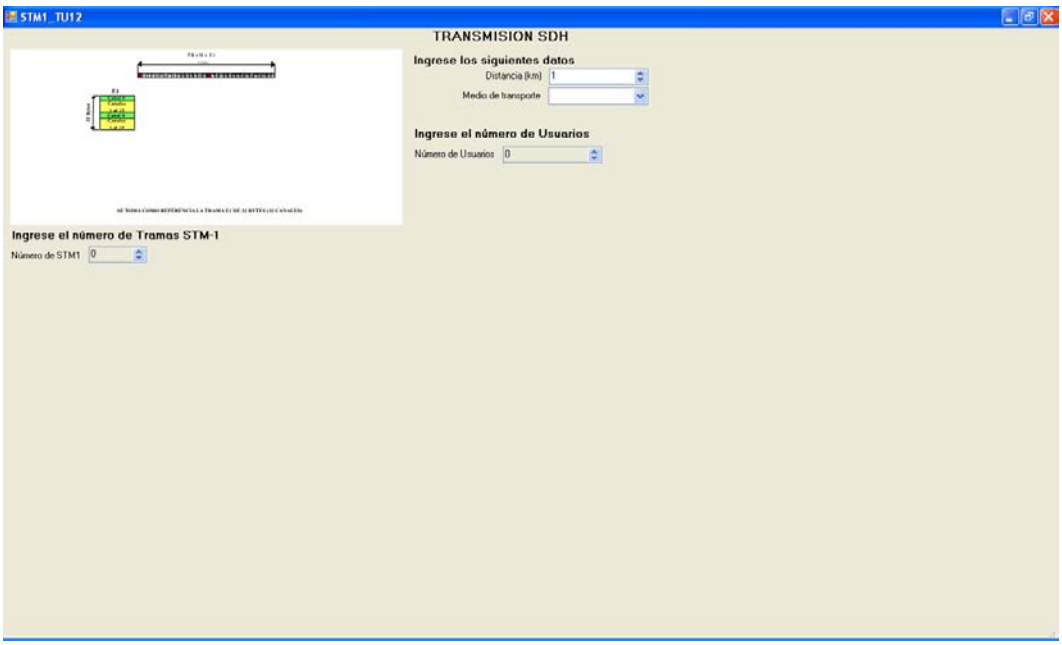


Fig. 5. 80 Interfaz de usuario para la selección de los parámetros de transmisión

El formulario consta de una sección para el ingreso de las variables que intervienen en la transmisión como: distancia y medio de transporte.

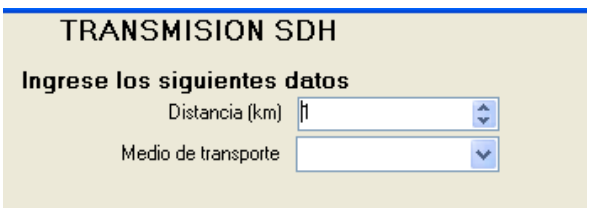


Fig. 5. 81 Ingreso de las variables que intervienen en la transmisión

El formulario tiene un espacio definido donde se visualizan las Tramas STM-1 y sus canales a los cuales se les asignarán un usuario dependiendo del ancho de banda seleccionado.

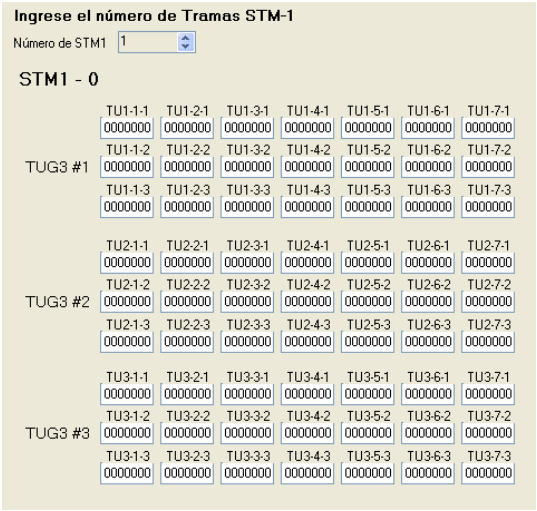


Fig. 5. 822 Formación de una trama STM-1 a nivel de TU-12

También dispone de un área para la interfaz de usuarios donde se debe seleccionar el Ancho de Banda y designar los canales de la Trama STM-1 para cada usuario (pestaña Transmisión) y seleccionar la información a codificarse y simularse su envío para después mostrar las estadísticas de la transmisión (pestaña Huffman).

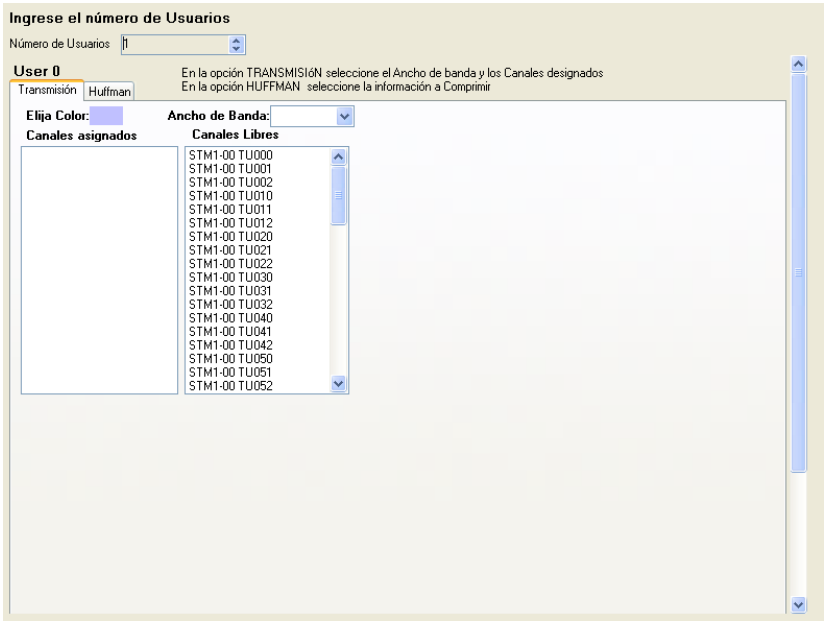


Fig. 5. 83 Interfaz de usuario interactuando con una trama STM-1 a nivel de canales TU-12

Se dispone de una sección donde se visualiza una animación ilustrativa que muestra como se forma una Trama STM-1 partiendo de Unidades Tributarias TU-12.

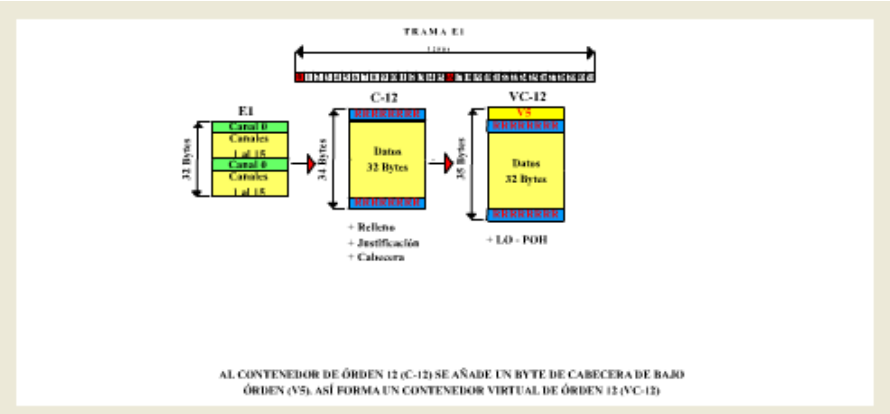


Fig. 5. 84 Animación que ejemplifica la formación de una trama STM-1 partiendo de un E1

Ingreso de las variable que intervienen en la transmisión.

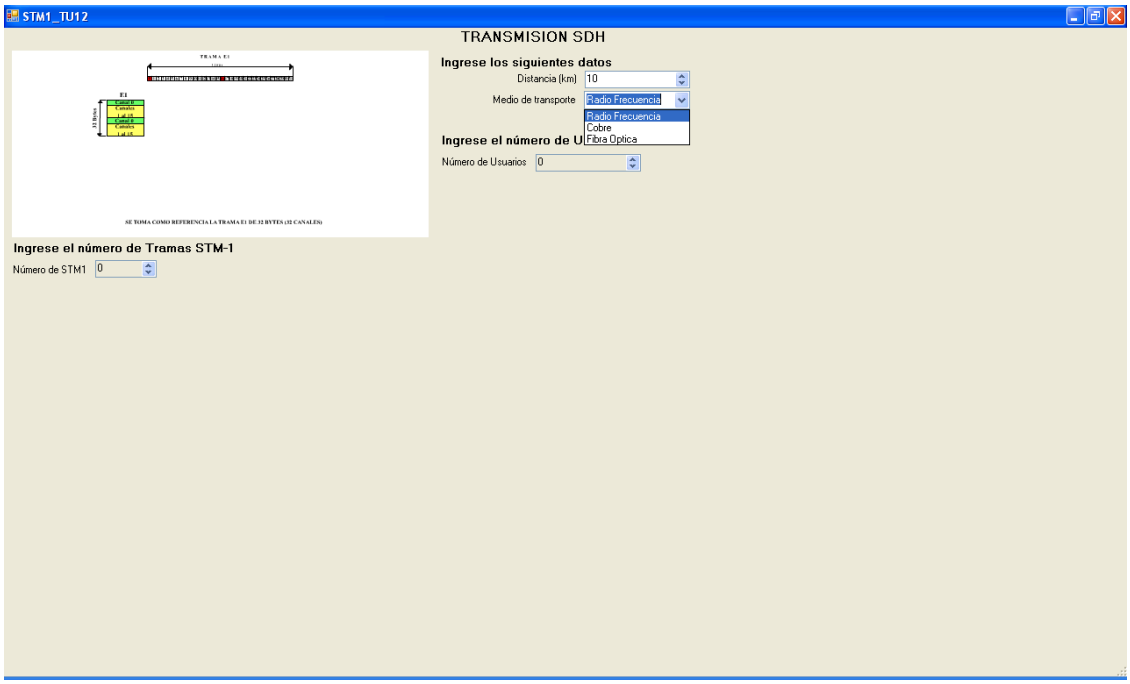


Fig. 5. 85 Ingreso de las variables que intervienen en la transmisión

Al elegir el número de Tramas E1 se carga el control que permite agregar Tramas E1 a nuestro formulario.

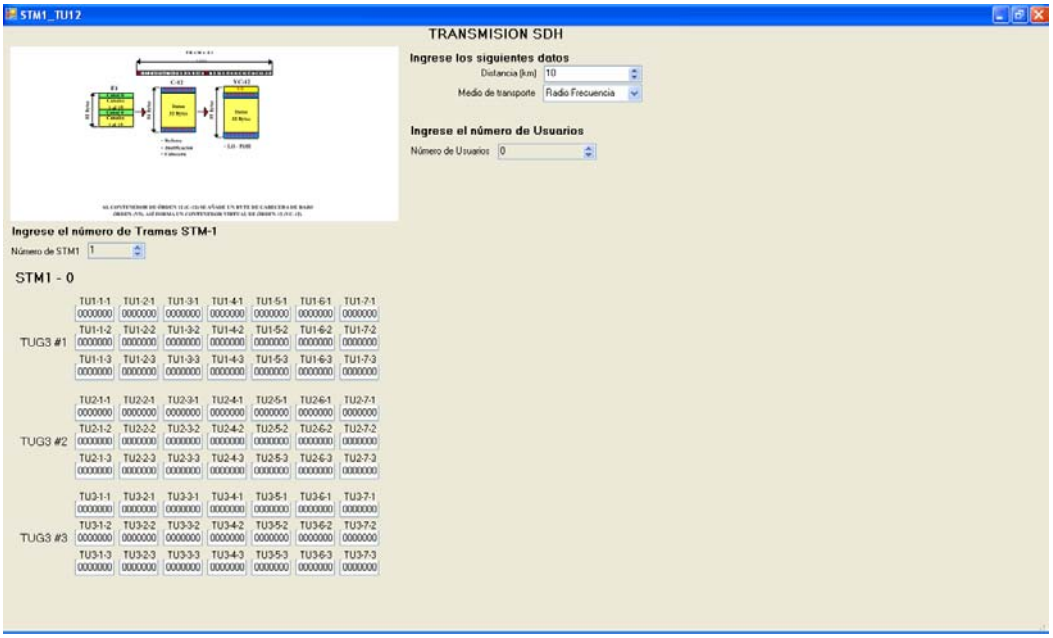


Fig. 5. 86 Se agregan los canales de una trama STM-1

Al incrementar el número de usuarios se carga la animación donde se visualiza la formación de la Trama E1 y se habilita las opciones de Transmisión y Huffman.

En la opción Transmisión se debe elegir el Ancho de Banda y el canal de la Trama E1 designado para el usuario. Para elegir el Ancho de Banda se selecciona de la lista.

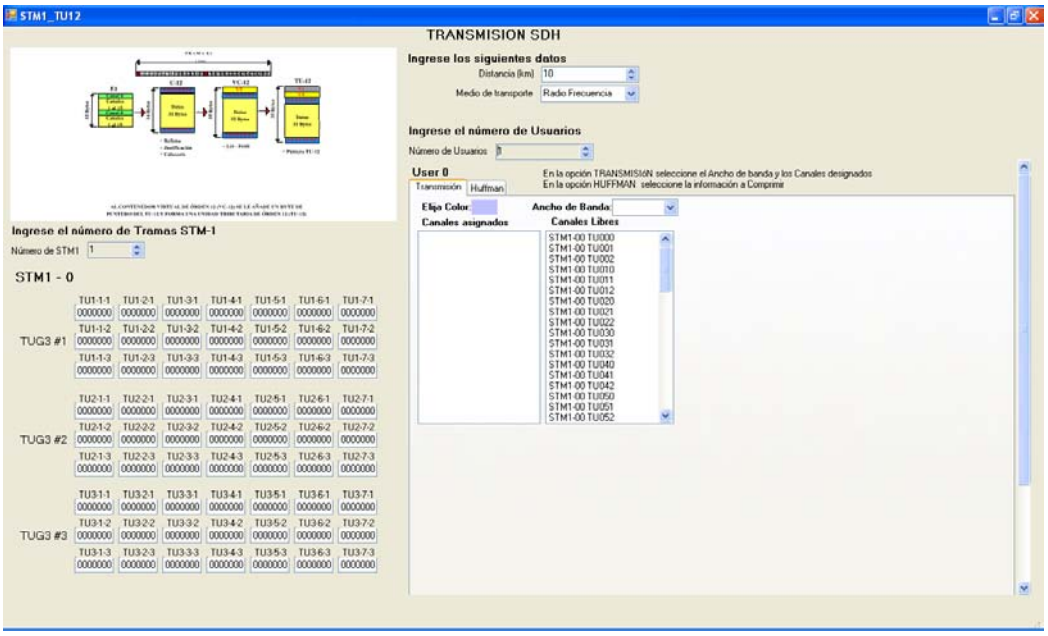


Fig. 5. 87 Elección del canal y los Anchos de Banda

Para elegir el canal se debe dar doble Click sobre el listado de Canales Disponibles, de este modo se visualizará el canal seleccionado pintado de diferente color para cada usuario. El listado de Canales Libres se actualiza de inmediato, con esto se controla que un usuario no elija un canal ocupado.

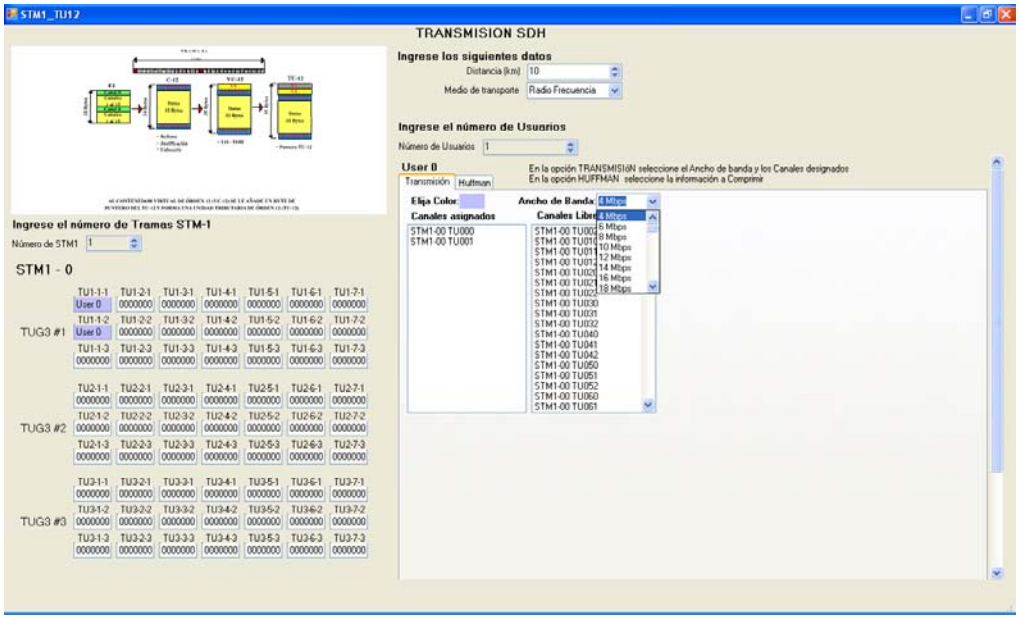


Fig. 5. 88 Canales de la trama STM-1 que han sido seleccionados

En la opción Huffman se debe elegir la información que se comprimirá y simulará el envío. Para ello Click en el botón Agregar y elegir los archivos a codificar.

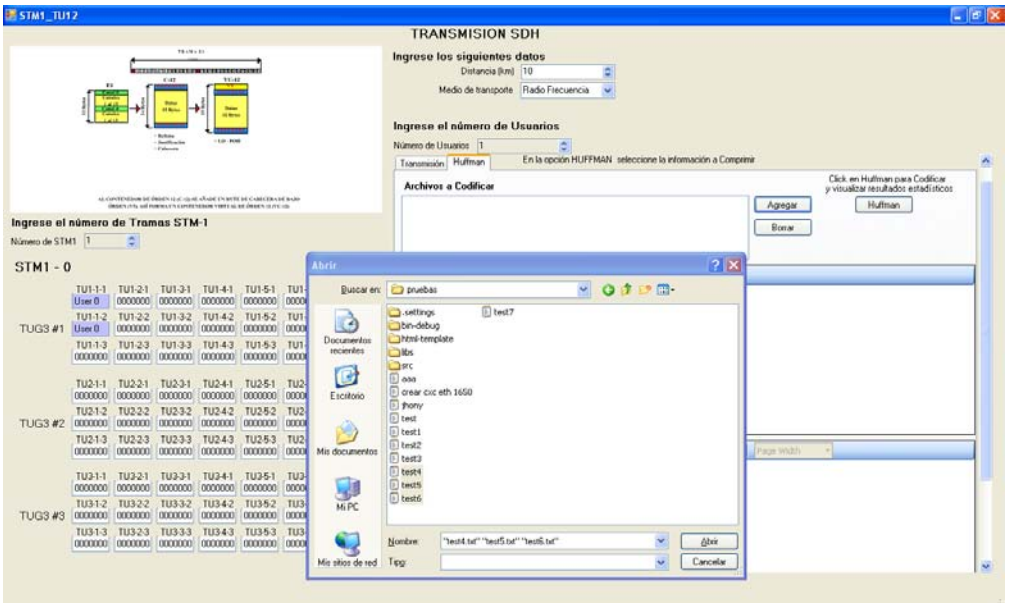


Fig. 5. 89 Elección de los archivos que intervienen en el proceso de la compresión

La información seleccionada se lista en el área de Archivos a Codificar, en esta área se puede Agregar o Borrar archivos a comprimir. Para borrar se debe seleccionar el archivo de la lista y luego pulsar el botón Borrar.

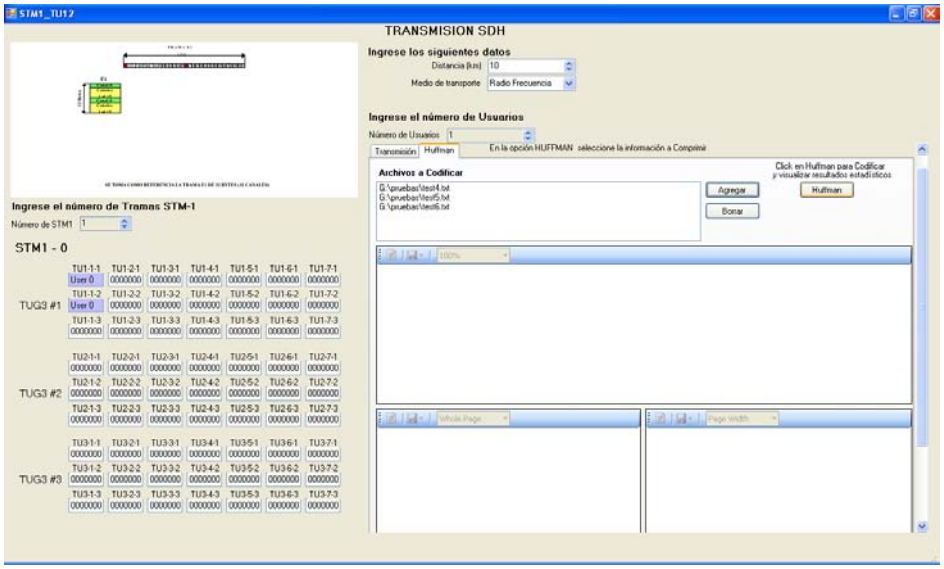


Fig. 5. 90 Vista de la interfaz de usuario, elección de archivos a comprimir

Una vez listada toda la información a codificar se procede a comprimirla. Para ello se debe hacer Click sobre el botón Huffman. Al hacer esta acción se visualizan las estadísticas de compresión y simulación de transmisión de los archivos seleccionados. Las estadísticas se muestran en forma de tabla y en forma gráfica.

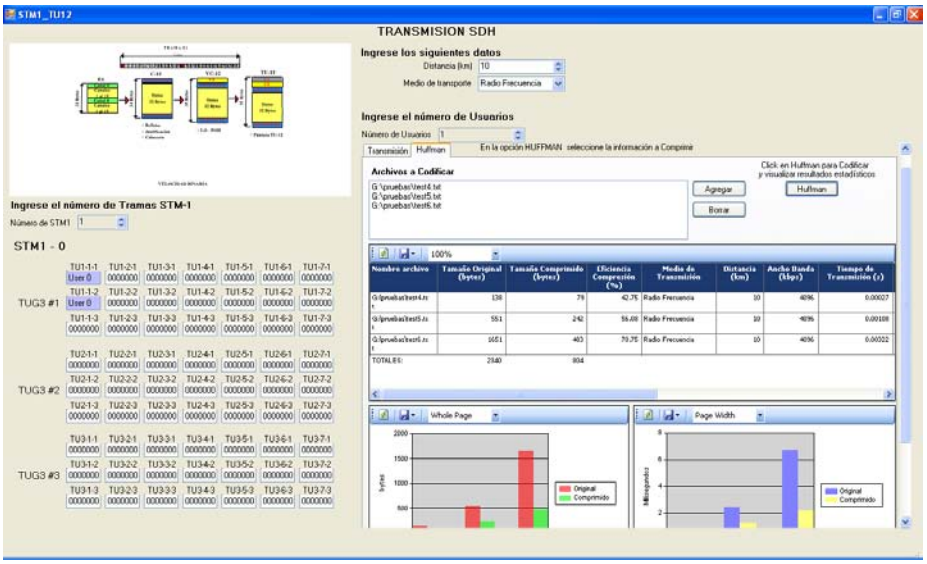


Fig. 5. 91 Visualización de las estadísticas de compresión

En el área de las estadísticas ya sea de forma de tabla y gráfico se tiene la posibilidad de exportar los datos a formato Excel y PDF.

Exportar los datos a Excel

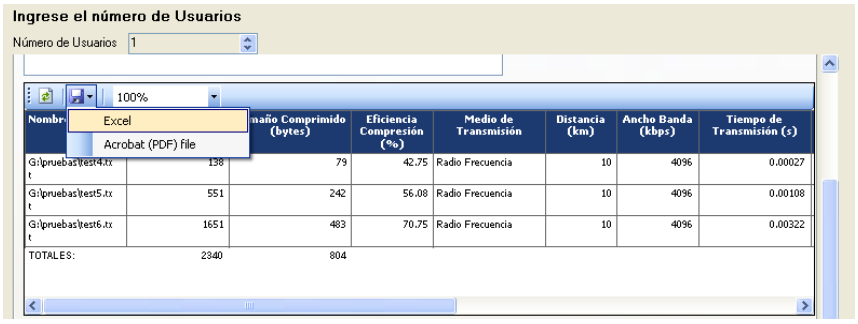


Fig. 5. 92 Se exporta a Excel las estadísticas obtenidas luego de la compresión

Exportar los datos a PDF

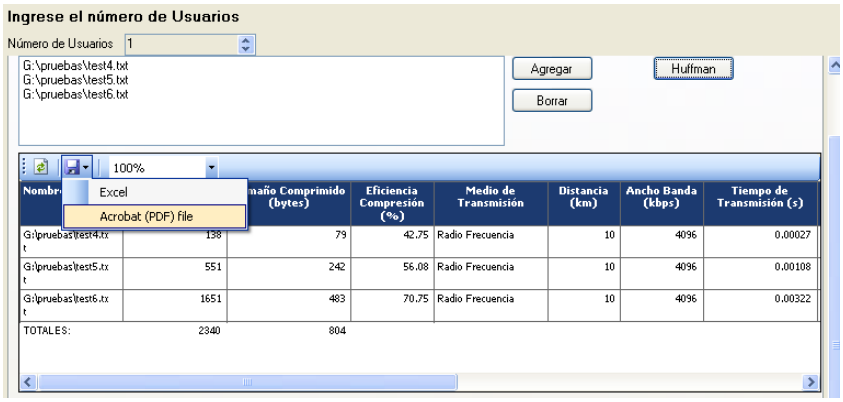


Fig. 5. 93 Se exporta a PDF las estadísticas obtenidas luego de la compresión

Resultados exportados a formato PDF

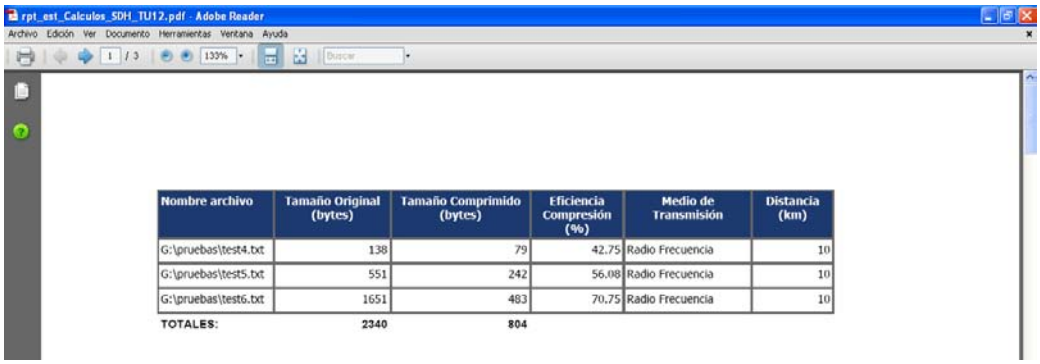


Fig. 5. 94 Resultados exportados a PDF

Resultados exportados a formato EXCEL

Fig. 5. 95 Resultados exportados a formato Excel

En todos los reportes se da la posibilidad de refrescar los datos.

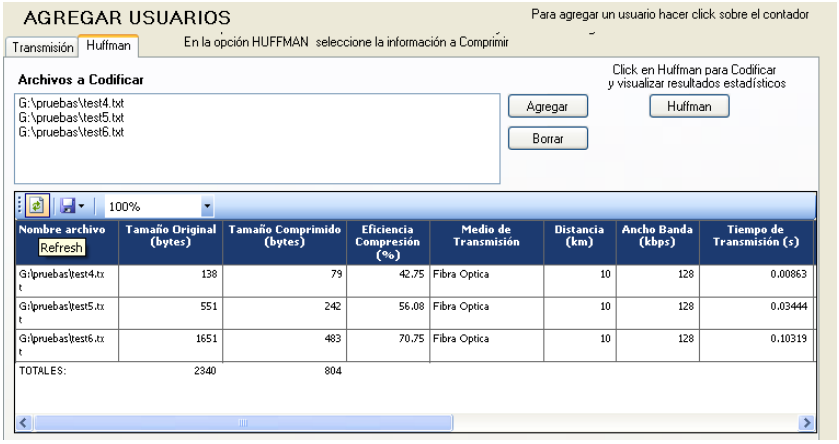


Fig. 5. 96 Refrescar los datos de la tabla

En todos los reportes se da la posibilidad de ampliar o disminuir la vista de acuerdo a las necesidades del cliente.

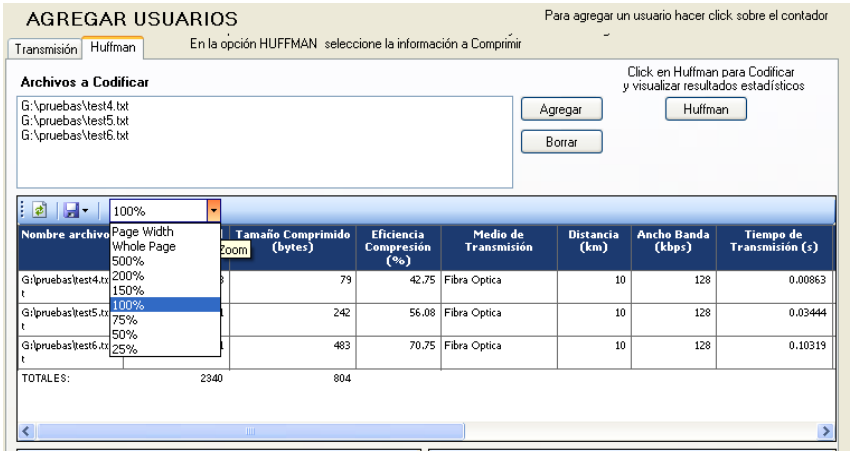


Fig. 5. 97 Ajustar el tamaño de visualización

Opción Varios Usuarios STM1 -> VC-3:

Muestra un formulario donde se simula el envío de información codificada de varios usuarios sobre varias tramas STM-1 en Contenedores Virtuales VC-3. Esta simulación permite aumentar Tramas STM-1 a medida que aumentan los usuarios de una red.

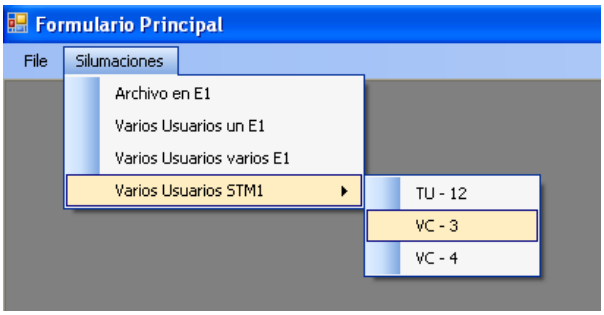


Fig. 5. 98 Opción varios Usuarios STM-1 interactuando con canales VC-3

Al seleccionar esta opción se despliega una ventana donde se tendrá que ingresar las variables que intervienen en la transmisión como: distancia, número de usuarios y medio de transporte.

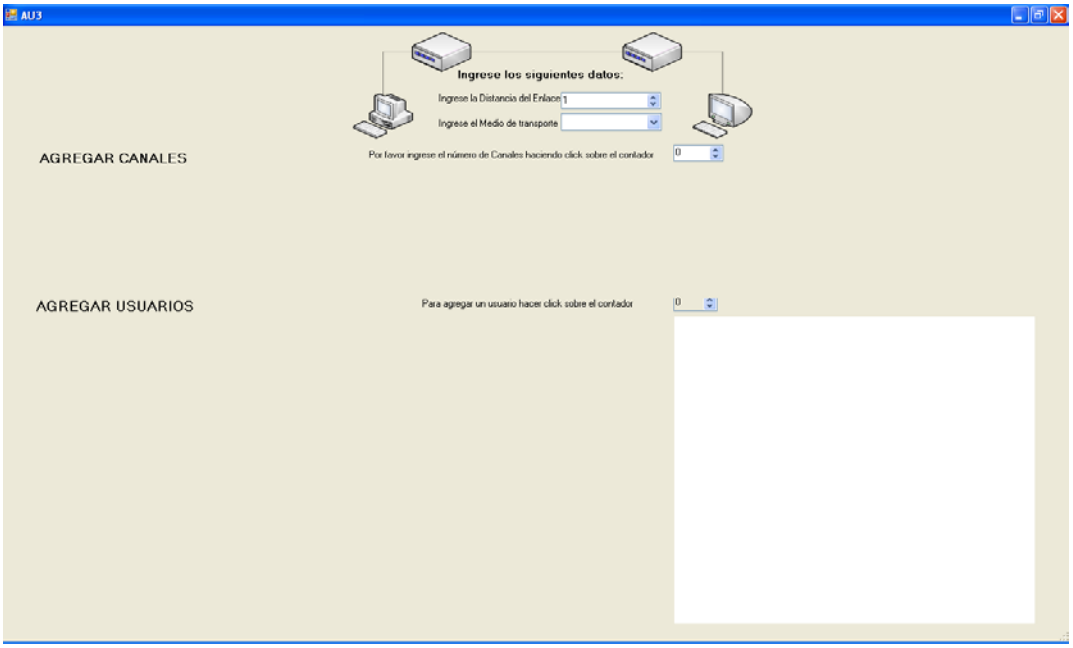


Fig. 5. 99 Interfaz de usuario para la selección de los parámetros de transmisión

El formulario consta de una sección para el ingreso de las variables que intervienen en la transmisión como: distancia y medio de transporte.

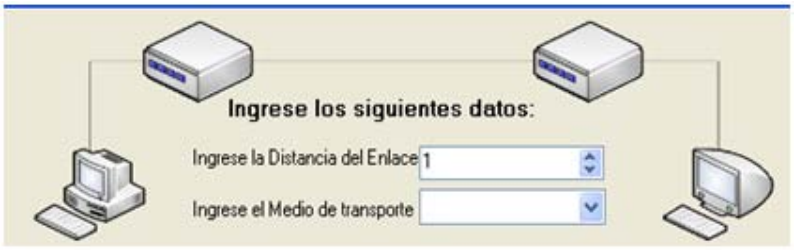


Fig. 5. 100 Ingreso de las variables que intervienen en la transmisión

El formulario tiene un espacio definido donde se visualizan las Tramas STM-1 y sus canales VC-3 a los cuales se les asignarán un usuario dependiendo del ancho de banda seleccionado.

Fig. 5. 101 Formación de una trama STM-1 dinámica

También dispone de un área para la interfaz de usuarios y una sección donde se visualiza una animación ilustrativa que muestra como se forma una Trama STM-1 a partir de Contenedores Virtuales VC-3.

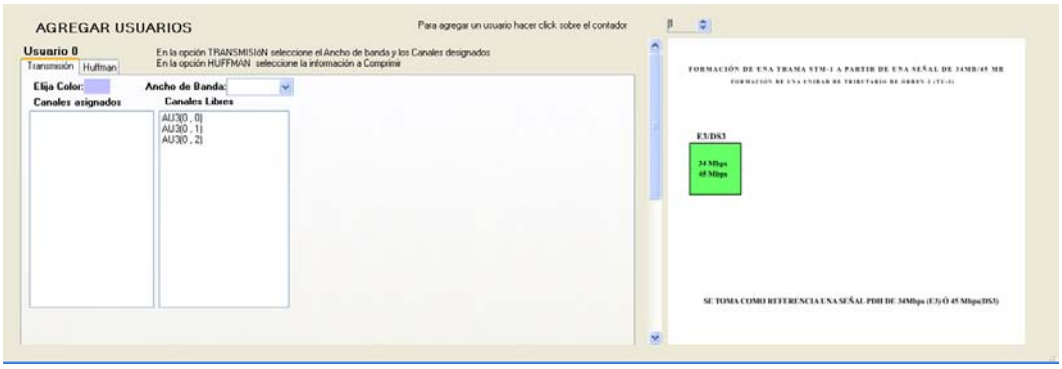


Fig. 5. 102 Selección de los canales que intervienen en la simulación

La interfaz de usuario dispone de dos pestañas Transmisión y Huffman.

La opción Transmisión permite elegir el Ancho de Banda y los canales de los usuarios.

La opción Huffman permite elegir la información que se codificará y simulará su envío, además se puede ver los resultados estadísticos del proceso de transmisión.

Ingreso de las variable que intervienen en la transmisión.

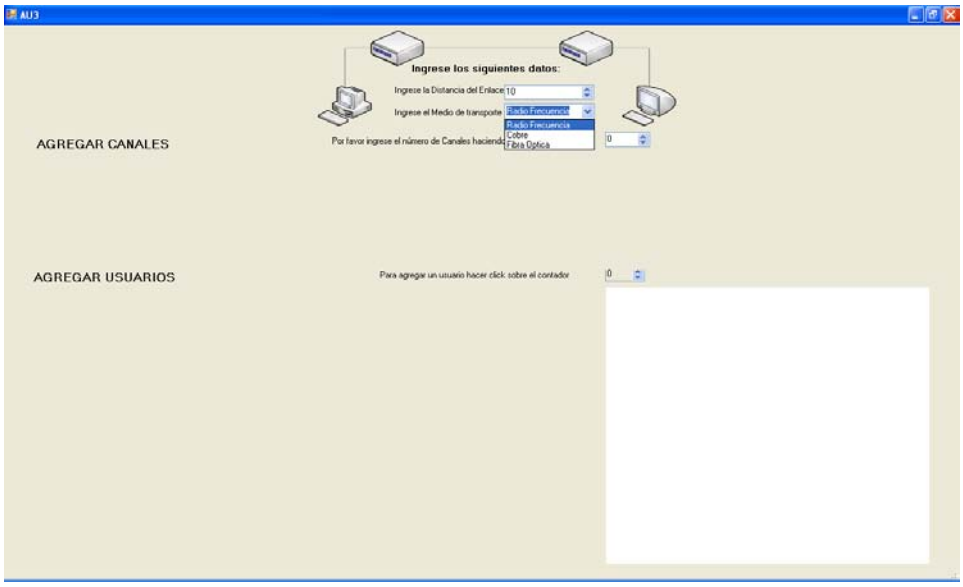


Fig. 5. 103 Selección de las variables de transmisión

Al elegir el número de Tramas STM-1 se carga el control que permite agregar Tramas STM-1 al formulario.

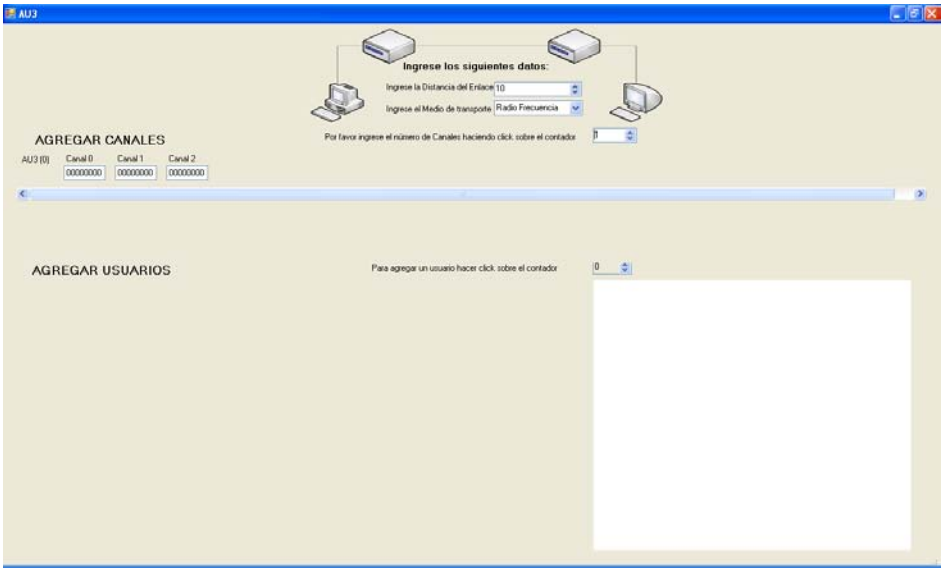


Fig. 5. 104 Agregar usuarios

Al incrementar el número de usuarios se carga la animación donde se visualiza la formación de la Trama E1 y se habilita las opciones de Transmisión y Huffman.

En la opción Transmisión se debe elegir el Ancho de Banda y el canal de la Trama E1 designado para el usuario.

Para elegir el Ancho de Banda se selecciona de la lista. Para elegir el canal se debe dar doble Click sobre el listado de Canales Disponibles, de este modo se visualizará el canal seleccionado pintado de diferente color para cada usuario.

El listado de Canales Libres se actualiza de inmediato, con esto se controla que un usuario no elija un canal ocupado.

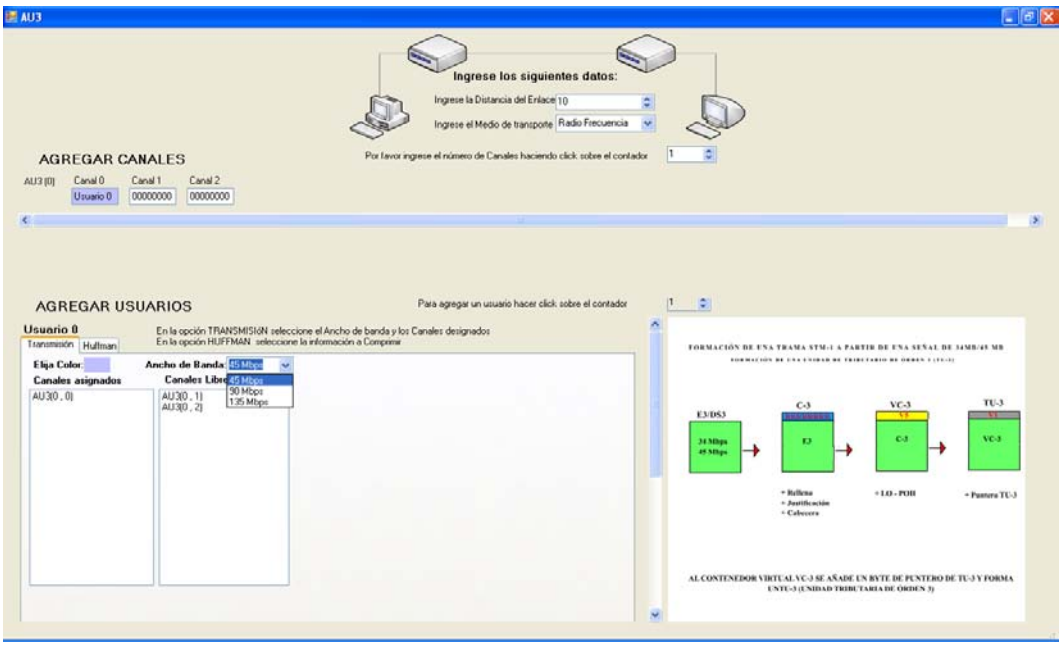


Fig. 5. 105 Selección de los canales del Ancho de Banda

En la opción Huffman se debe elegir la información que se comprimirá y simulará el envío. Para ello Click en el botón Agregar y elegir los archivos a codificar.

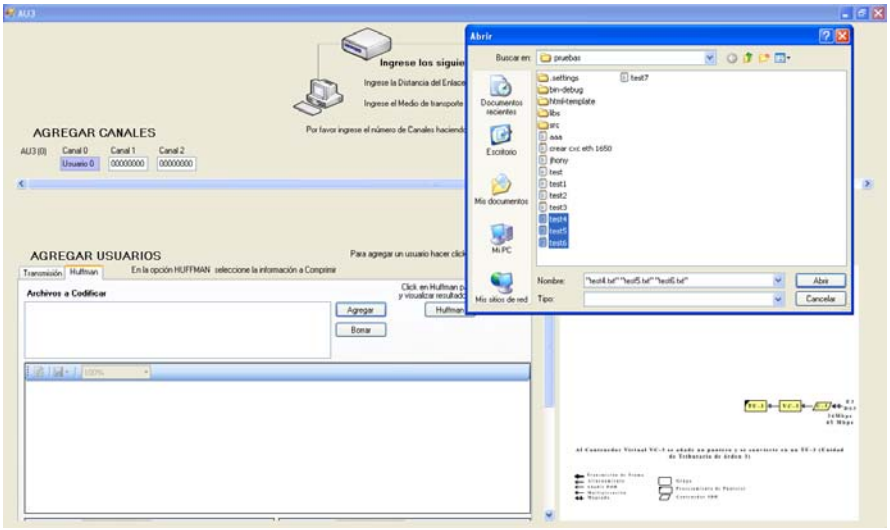


Fig. 5. 106 Elección de los archivos que intervienen en el proceso de la compresión

La información seleccionada se lista en el área de Archivos a Codificar, en esta área se puede Agregar o Borrar archivos a comprimir. Para borrar se debe seleccionar el archivo de la lista y luego pulsar el botón Borrar.

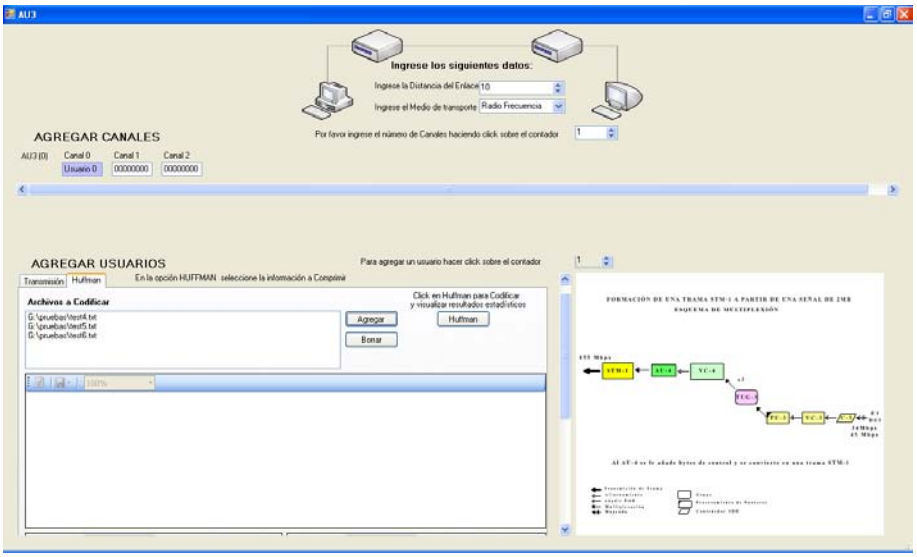


Fig. 5. 107 Vista de la interfaz de usuario, elección de archivos a comprimir

Una vez listada toda la información a codificar se procede a comprimirla. Para ello se debe hacer Click sobre el botón Huffman. Al hacer esta acción se visualizan las estadísticas de compresión y simulación de transmisión de los archivos seleccionados.

Las estadísticas se muestran en forma de tabla y en forma gráfica.

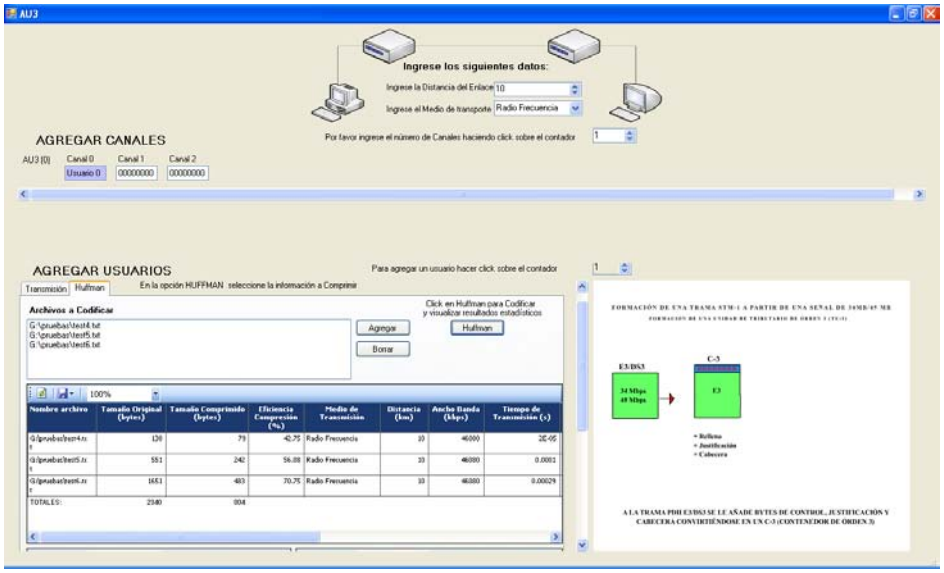


Fig. 5. 108 Estadísticas de compresión

En el área de las estadísticas ya sea de forma de tabla y gráfico se tiene la posibilidad de exportar los datos a formato Excel y PDF.

Exportar los datos a Excel

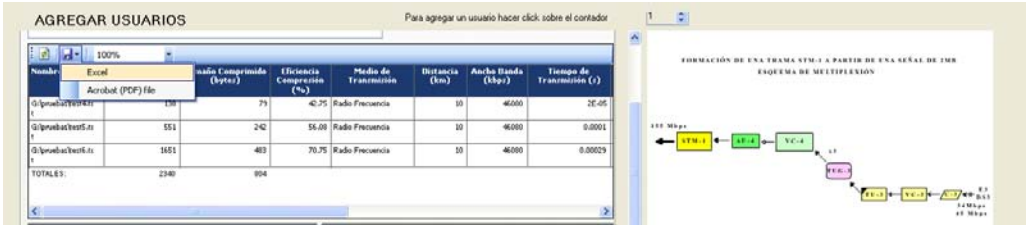


Fig. 5. 109 Se exporta a Excel las estadísticas obtenidas luego de la compresión

Exportar los datos a PDF

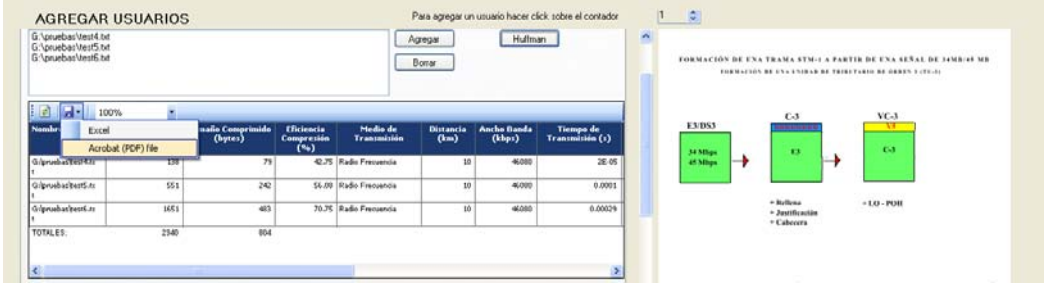
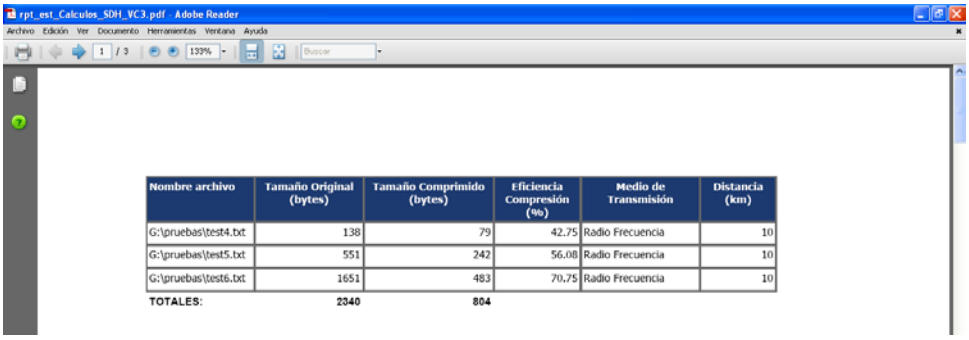


Fig. 5. 110 Se exporta a PDF las estadísticas obtenidas luego de la compresión

Resultados exportados a formato PDF



Nombre archivo	Tamaño Original (bytes)	Tamaño Comprimido (bytes)	Eficiencia Compresión (%)	Medio de Transmisión	Distancia (km)
G:\pruebas\test4.txt	138	79	42.75	Radio Frecuencia	10
G:\pruebas\test5.txt	551	242	56.08	Radio Frecuencia	10
G:\pruebas\test6.txt	1651	483	70.75	Radio Frecuencia	10
TOTALES:	2340	804			

Fig. 5. 111 Se exporta a PDF las estadísticas obtenidas luego de la compresión

Resultados exportados a formato EXCEL

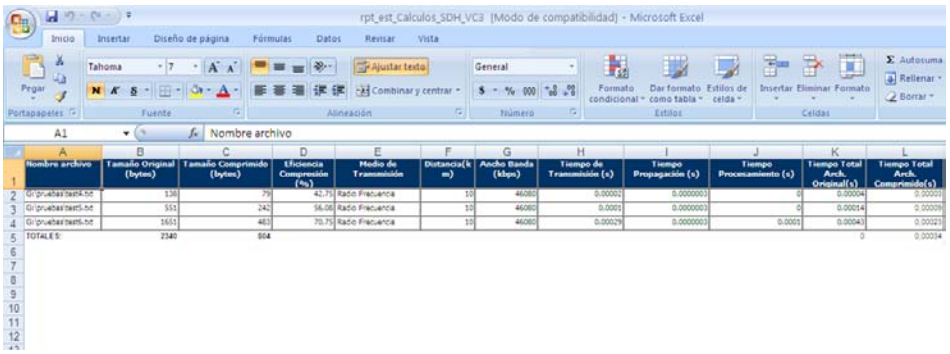
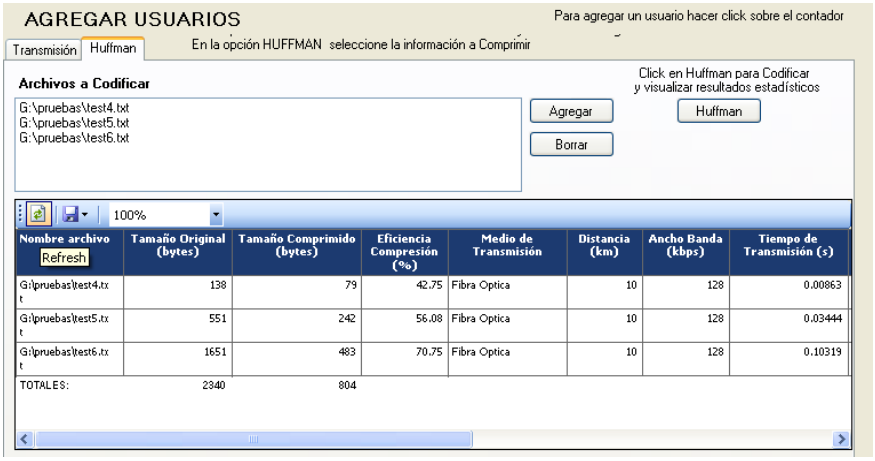


Fig. 5. 112 Resultados exportados a formato Excel

En todos los reportes se da la posibilidad de refrescar los datos.



Nombre archivo	Tamaño Original (bytes)	Tamaño Comprimido (bytes)	Eficiencia Compresión (%)	Medio de Transmisión	Distancia (km)	Ancho Banda (kbps)	Tiempo de Transmisión (s)
G:\pruebas\test4.txt	138	79	42.75	Fibra Optica	10	128	0.00863
G:\pruebas\test5.txt	551	242	56.08	Fibra Optica	10	128	0.03444
G:\pruebas\test6.txt	1651	483	70.75	Fibra Optica	10	128	0.10319
TOTALES:	2340	804					

Fig. 5. 113 Refrescar los datos de la tabla

En todos los reportes se da la posibilidad de ampliar o disminuir la vista de acuerdo a las necesidades del cliente.

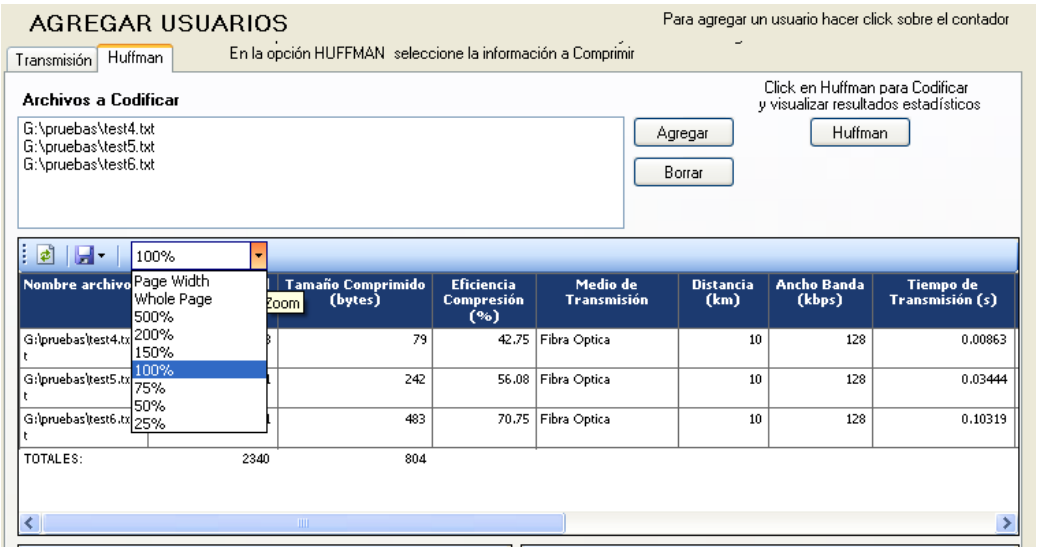


Fig. 5. 114 Ajustar el tamaño de visualización

Opción Varios Usuarios STM1 -> VC-4:

Muestra un formulario donde se simula el envío de información codificada de varios usuarios sobre varias tramas STM-1 en Contenedores Virtuales VC-4. Esta simulación permite aumentar Tramas STM-1 a medida que aumentan los usuarios de una red.

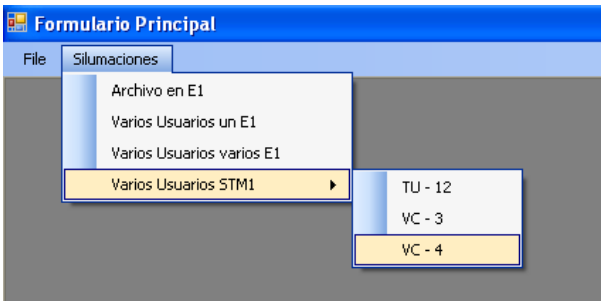


Fig. 5. 115 Opción varios Usuarios interactuando con varios VC-4

Al seleccionar esta opción se despliega una ventana donde se tendrá que ingresar las variables que intervienen en la transmisión como: distancia, número de usuarios y medio de transporte.

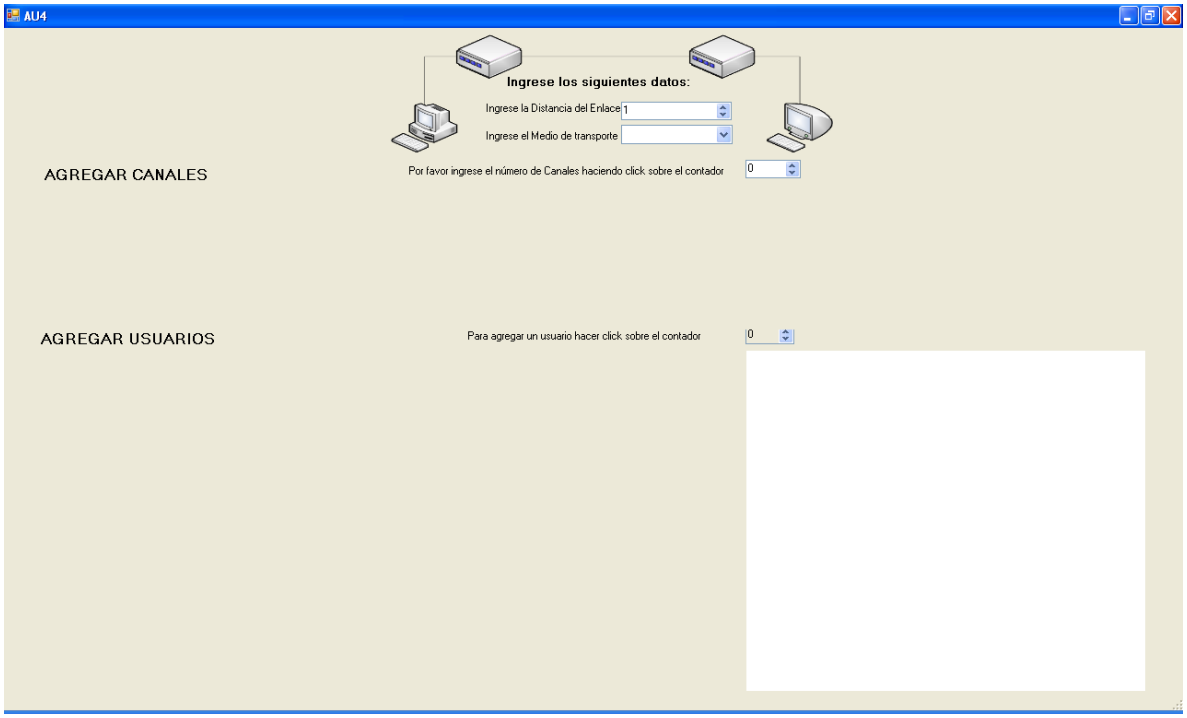


Fig. 5. 116 Interfaz de usuario para la selección de los parámetros de transmisión

El formulario consta de una sección para el ingreso de las variables que intervienen en la transmisión como: distancia y medio de transporte.

Fig. 5. 117 Ingreso de las variables que intervienen en la transmisión

El formulario tiene un espacio definido donde se visualizan las Tramas STM-1 y sus canales VC-4 a los cuales se les asignarán un usuario dependiendo del ancho de banda seleccionado.

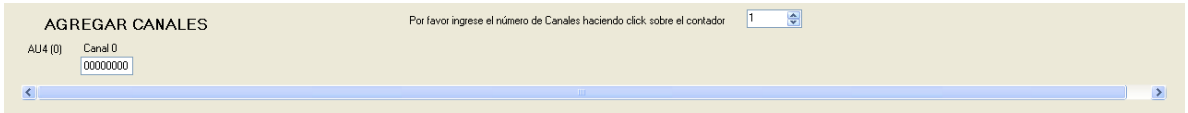


Fig. 5. 118 Formación de una trama STM-1 a nivel de TU-12

También dispone de un área para la interfaz de usuarios y una sección donde se visualiza una animación ilustrativa que muestra como se forma una Trama STM-1 a partir de Contenedores Virtuales VC-4.

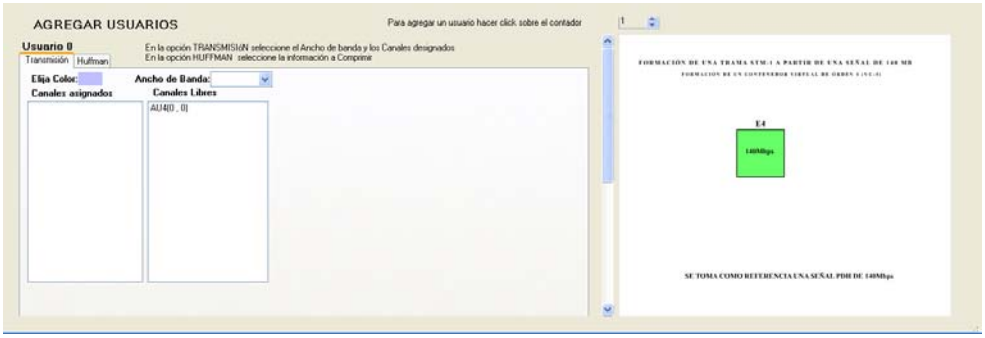


Fig. 5. 119 Interfaz de usuario interactuando con una trama STM-1 a nivel de canales VC-4

La interfaz de usuario dispone de dos pestañas Transmisión y Huffman.

La opción Transmisión permite elegir el Ancho de Banda y los canales de los usuarios. La opción Huffman permite elegir la información que se codificará y simulará su envío, además se puede ver los resultados estadísticos del proceso de transmisión.

Ingreso de las variable que intervienen en la transmisión.

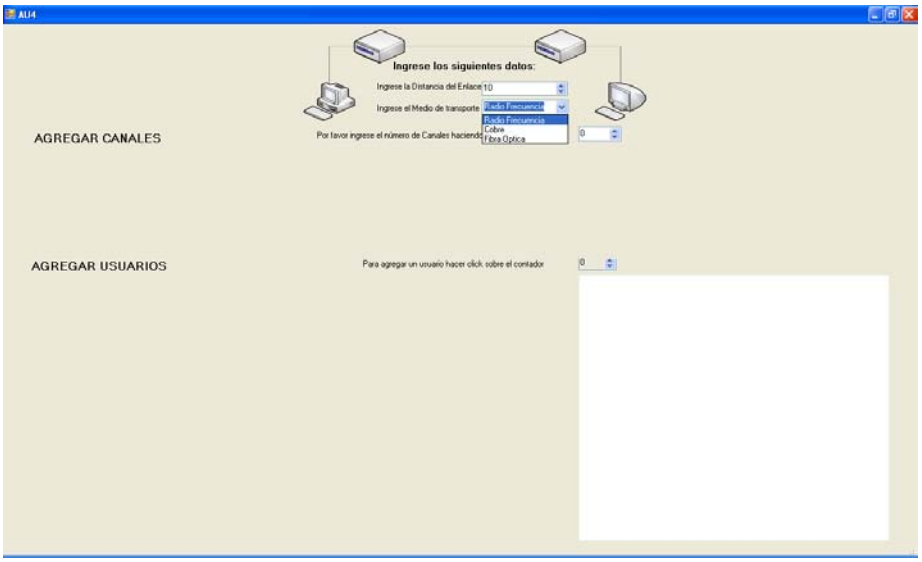


Fig. 5. 120 Ingreso de las variables que intervienen en la transmisión

Al elegir el número de Tramas STM-1 se carga el control que permite agregar Tramas STM-1 al formulario.

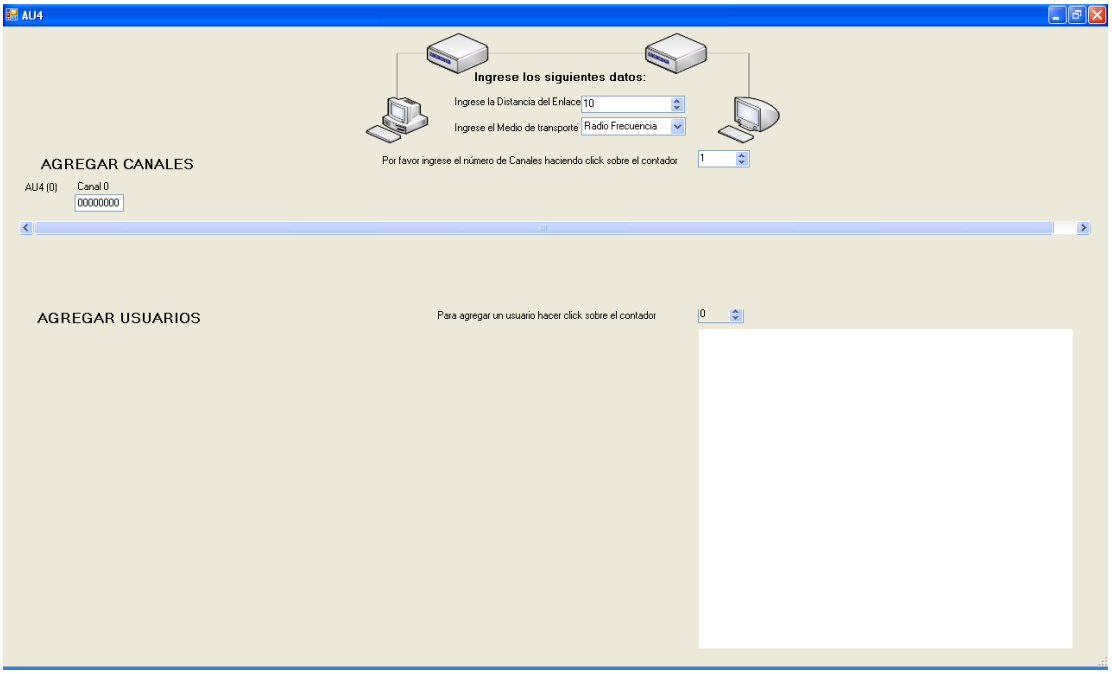


Fig. 5. 121 Se agregan los canales de una trama STM-1

Al incrementar el número de usuarios se carga la animación donde se visualiza la formación de la Trama STM-1 y se habilita las opciones de Transmisión y Huffman.

En la opción Transmisión se debe elegir el Ancho de Banda y el canal de la Trama E1 designado para el usuario. Para elegir el Ancho de Banda se selecciona de la lista.

Para elegir el canal se debe dar doble Click sobre el listado de Canales Disponibles, de este modo se visualizará el canal seleccionado pintado de diferente color para cada usuario.

El listado de Canales Libres se actualiza de inmediato, con esto se controla que un usuario no elija un canal ocupado.

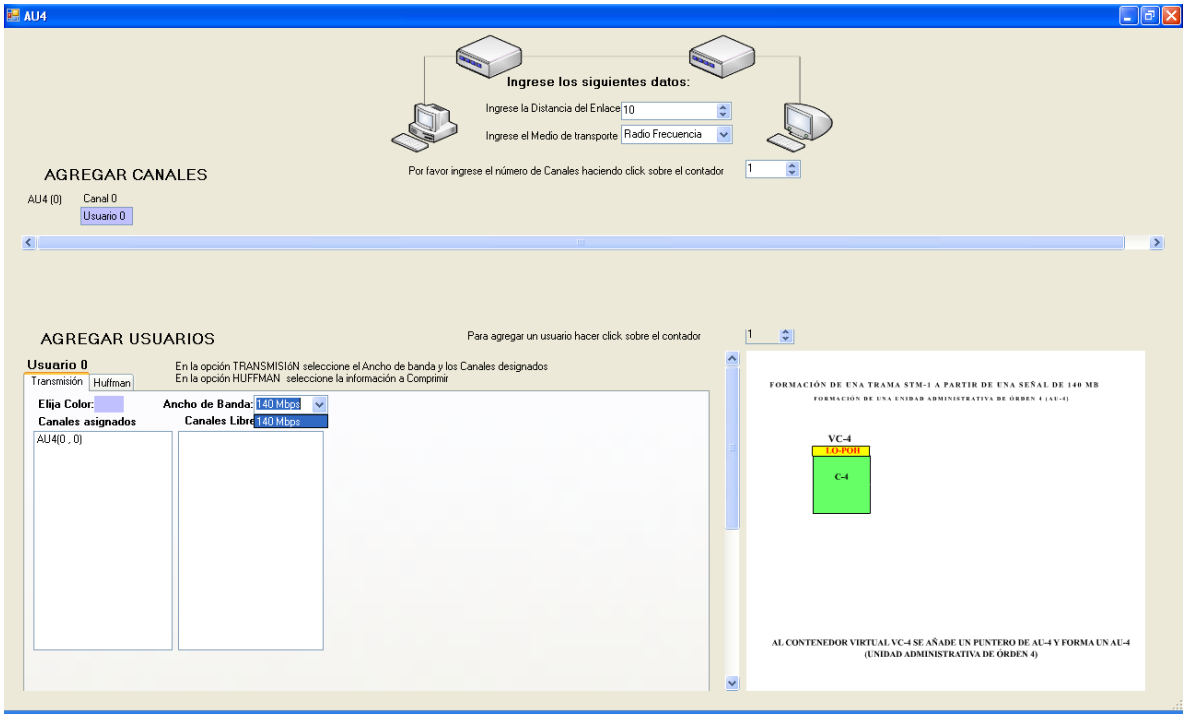


Fig. 5. 122 Elección del canal y los Anchos de Banda

En la opción Huffman se debe elegir la información que se comprimirá y simulará el envío. Para ello Click en el botón Agregar y elegir los archivos a codificar.

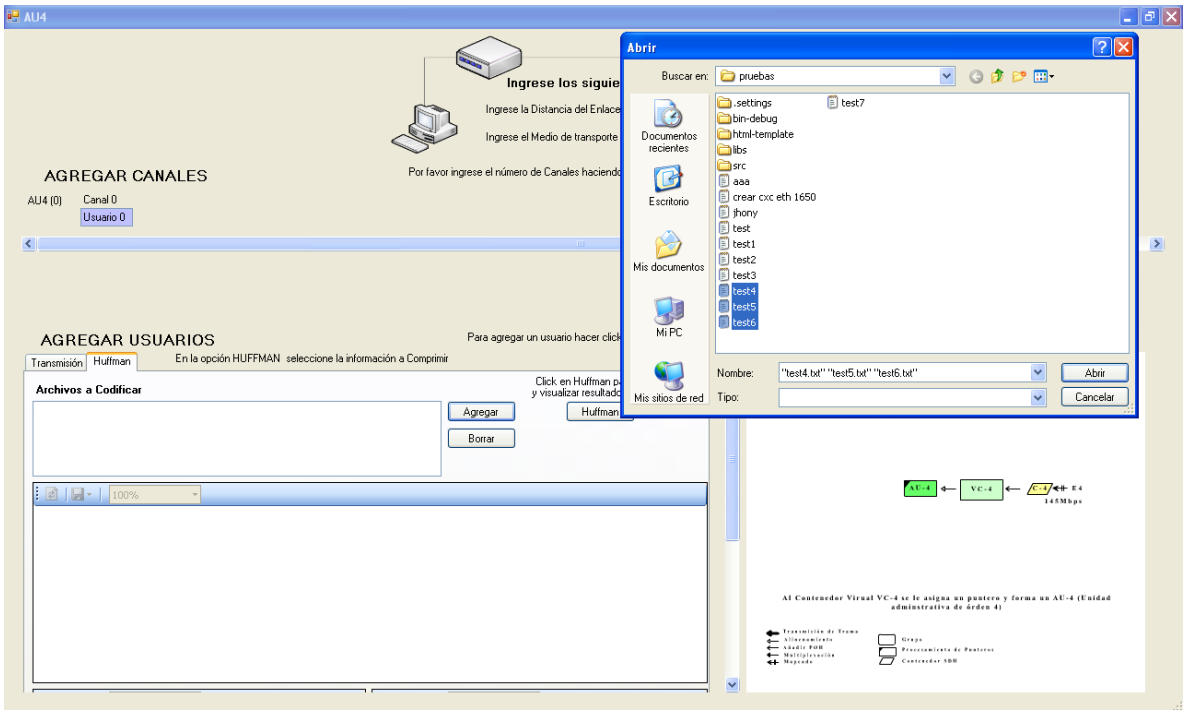


Fig. 5. 123 Elección de los archivos que intervienen en el proceso de la compresión

La información seleccionada se lista en el área de Archivos a Codificar, en esta área se puede Agregar o Borrar archivos a comprimir. Para borrar se debe seleccionar el archivo de la lista y luego pulsar el botón Borrar.

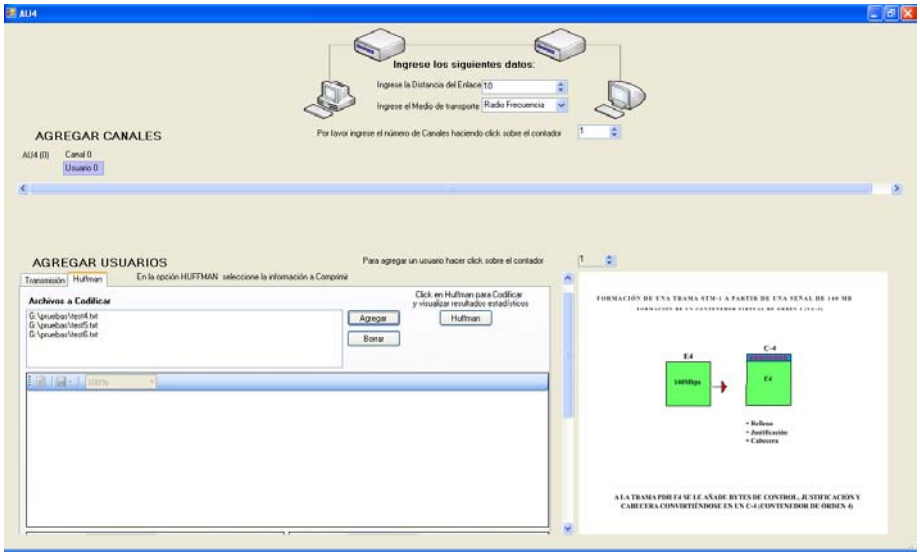


Fig. 5. 124 Vista de la interfaz de usuario, elección de archivos a comprimir

Una vez listada toda la información a codificar se procede a comprimirla. Para ello se debe hacer Click sobre el botón Huffman. Al hacer esta acción se visualizan las estadísticas de compresión y simulación de transmisión de los archivos seleccionados. Las estadísticas se muestran en forma de tabla y en forma gráfica.

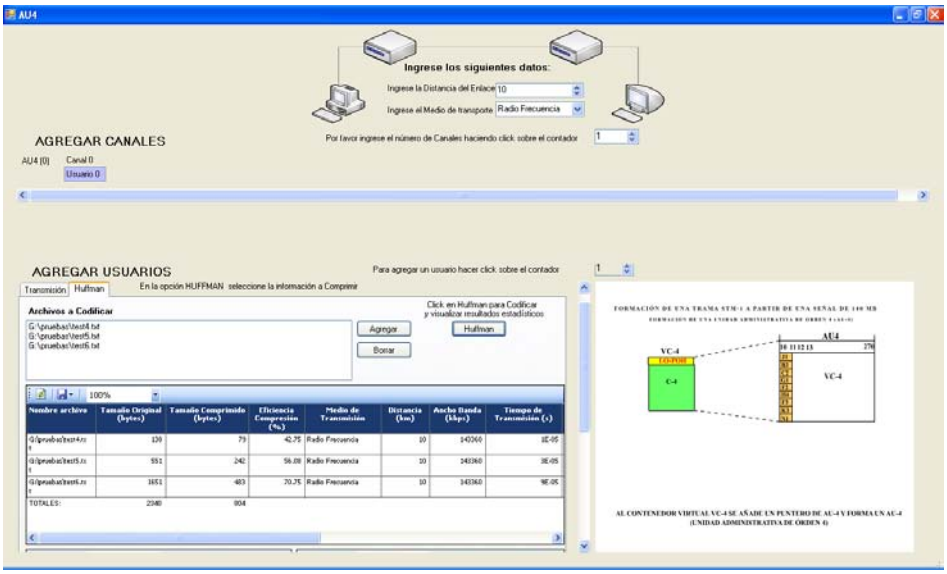


Fig. 5. 125 Visualización de las estadísticas de compresión

En el área de las estadísticas ya sea de forma de tabla y gráfico se tiene la posibilidad de exportar los datos a formato Excel y PDF.

Exportar los datos a Excel

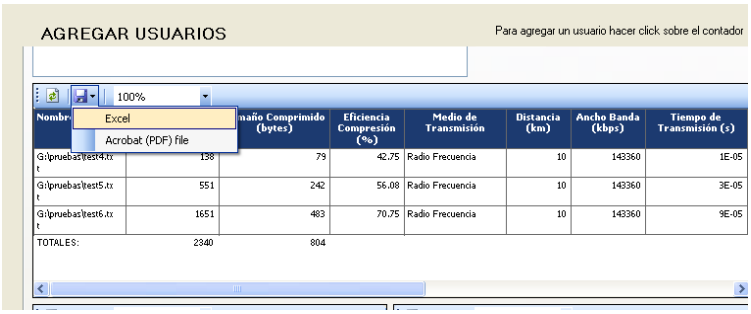


Fig. 5. 126 Se exporta a Excel las estadísticas obtenidas luego de la compresión

Exportar los datos a PDF

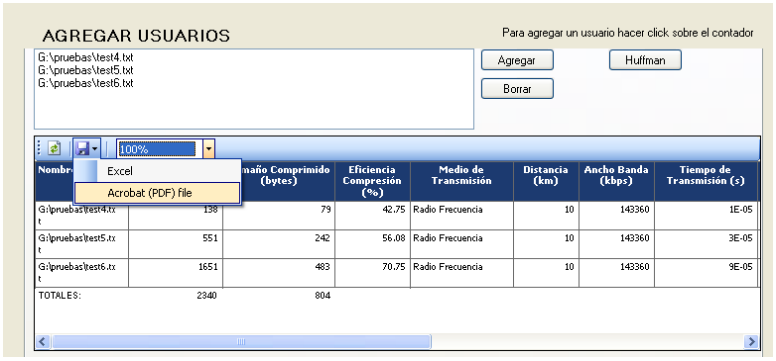


Fig. 5. 127 Se exporta a PDF las estadísticas obtenidas luego de la compresión

Resultados exportados a formato PDF

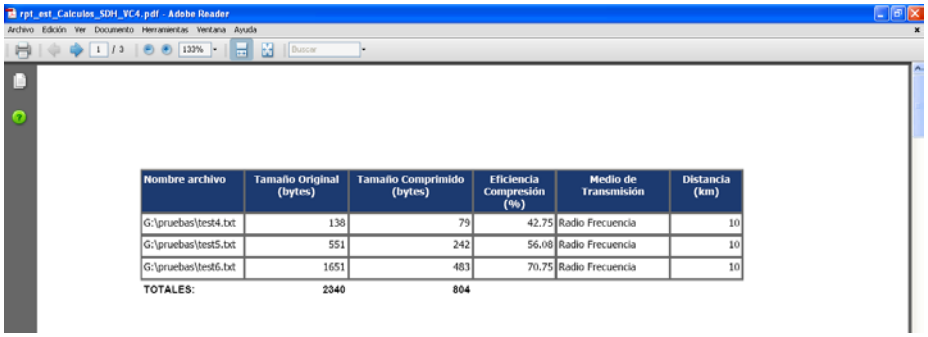


Fig. 5. 128 Resultados exportados a PDF

Resultados exportados a formato EXCEL

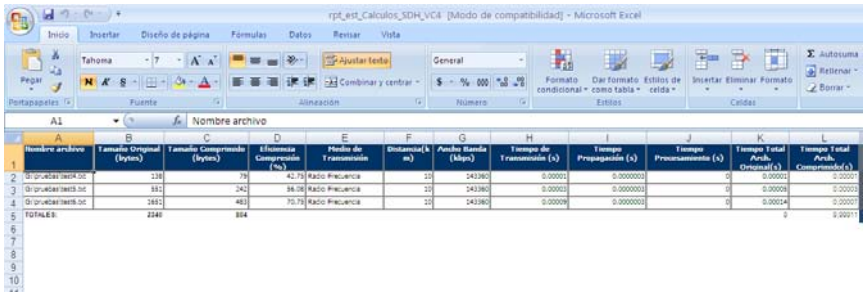


Fig. 5. 129 Resultados exportados a formato Excel

En todos los reportes se da la posibilidad de refrescar los datos.

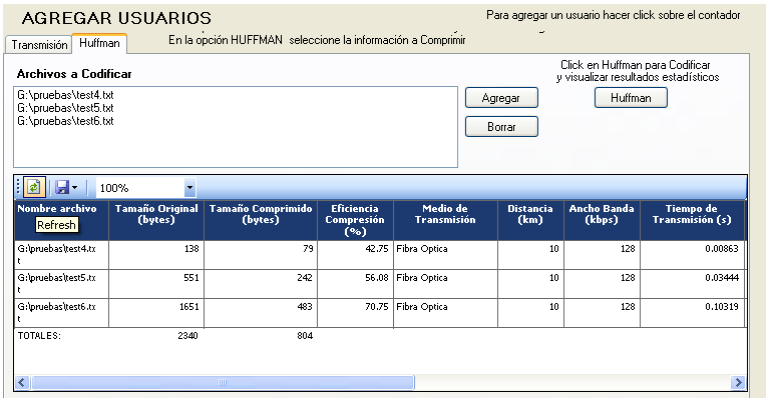


Fig. 5. 130 Refrescar los datos de la tabla

En todos los reportes se da la posibilidad de ampliar o disminuir la vista de acuerdo a las necesidades del cliente.

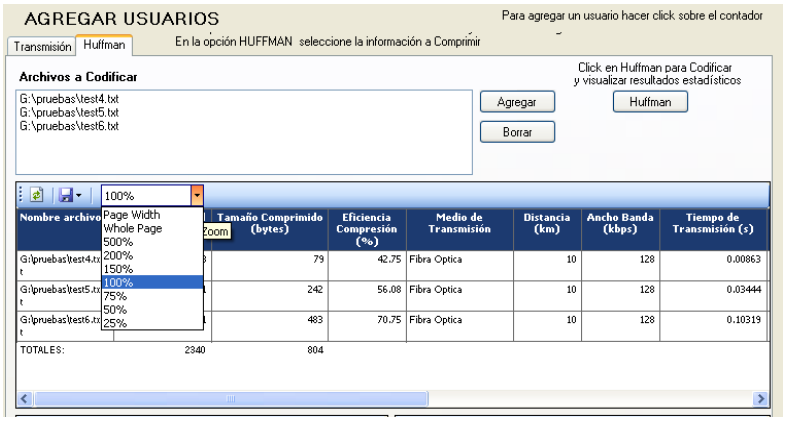


Fig. 5. 131 Ajustar el tamaño de visualización

6. CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

- El amplio uso de las redes Ethernet se justifica, en primer lugar, por costos bajos en crecimiento y explotación y en segundo lugar, por su flexibilidad a soportar nuevos y mejores servicios, que sigan posibilitando el mantenimiento de los ritmos de crecimiento que éste está experimentando. Por otra parte las tecnologías de transmisión TDM como PDH y SDH minimizan los costos al cliente final gracias a la multiplexación de varias señales de distintos clientes por el mismo medio físico.

- PDH básicamente fue diseñado para el tráfico de voz que fue predominante en las redes de telecomunicaciones cuyo principal servicio era la telefonía tradicional. Sin embargo, con el pasar de los años y debido al aparecimiento del Internet, surgió un crecimiento asombroso del tráfico de datos a través de las redes, lo que ocasionó una gran demanda de comunicaciones de alta velocidad, de esta manera PDH ha dado paso al desarrollo de nuevas tecnologías de transporte que cumplan las necesidades de los clientes, en especial los clientes móviles.

- Los patrones de tráfico hasta ahora han cambiado drásticamente, actualmente el tráfico de datos supera ampliamente al tráfico de voz y es por esa razón que PDH ha pasado a ser de una tecnología de transmisión troncal a una tecnología de acceso, siendo reemplazada por tecnologías de alta velocidad como SDH y DWDM. Así mismo se ha sustituido el par de cobre por fibra óptica en lo que respecta al medio físico de propagación.

- Las tecnologías PDH y SDH, se basan en multiplexores digitales que, mediante técnicas de multiplexación por división de tiempo, permiten

combinar varias señales digitales de jerarquía inferior en una señal digital de velocidad superior. Entre las nuevas tecnologías de transmisión en surgir se encuentra DWDM, caracterizada por sus altas capacidades de transmisión, aprovechando el multiplexado de diferentes longitudes de onda.

- SDH es una de las tecnologías de transporte dominante en las redes metropolitanas de los proveedores de servicios de telecomunicaciones, prácticamente todas las operadoras telefónicas de Ecuador disponen en su red de enlaces SDH formando anillos que manejan velocidades desde STM-1 hasta STM-64, algunas de estas redes SDH ya están manejando tráfico Ethernet y actualmente brindan servicios ADSL utilizando como medio de transporte su infraestructura SDH.

- La tecnología SDH se diseñó inicialmente para redes troncales, por lo que posee mecanismos robustos de disponibilidad y fiabilidad basados en un alto nivel de redundancia y tolerancia a fallos. En cambio, la tecnología Ethernet se diseñó principalmente para redes de empresa donde los requisitos de disponibilidad no son tan altos.

- La compresión de datos consiste en reducir el tamaño físico de cierta información. Un compresor se vale de un algoritmo que optimiza los datos, teniendo en cuenta el tipo de dato que se va a comprimir. El algoritmo de compresión seleccionado para este trabajo se basa en los conceptos de un algoritmo reversible de compresión de datos, resulta muy eficaz en la compresión de texto, ya que utiliza un análisis de frecuencias para conseguir su objetivo.

- Dentro de los estándares de compresión más utilizados para el tratamiento de voz se manejan los siguientes; G711, G728 y G729, esta compresión se lleva a cabo a nivel de capa física y en general son efectuados mediante hardware. Mientras que para VoIP se maneja H323 que es un protocolo estándar para aplicaciones de audio y video en entornos IP. Por otro lado,

los estándares más comunes para compresión de datos de aplicaciones de audio y video son; MP3, MP4, PMEG, JPEG, H.263, esta compresión se la realiza en el paso de la capa Aplicación a la capa Presentación

- El proceso de compresión de datos aprovecha el ancho de banda e incrementa la transferencia de información al reducir el tamaño de las tramas y del tiempo de transmisión de las mismas, lo que permite que se transmitan más datos a través de los enlaces dispuestos para ello. Esto permite aumentar el rendimiento y la disponibilidad del servicio para los usuarios finales sin tener que realizar grandes inversiones en infraestructura adicional.

- Para la realización del presente algoritmo de comprensión, se debe estar familiarizado con ciertas estructuras de datos y métodos de manipulación como; streams readers, arrays, listas encadenadas y árboles binarios, pues todos ellos intervienen en el proceso de compresión que se presenta en el siguiente trabajo.

- La aplicación de software se ajusta a un perfil didáctico, es por ello que se ha incorporado animaciones que ayuden a comprender el funcionamiento básico de estas tecnologías. También busca proyectar un ambiente simple y sencillo donde quien lo ejecute comprenda con facilidad la formación de las tramas PDH y SDH, la asignación de la información dentro de las tramas y las ventajas de disponer de mecanismos de compresión de información dentro de una red, como es el minimizar los tiempos de transmisión de la información.

- En la aplicación se simula un enlace punto a punto ya sea PDH o SDH, por el cual se determinan los tiempos que toma transmitir un archivo normal y un archivo comprimido. De esta manera se comprueba mediante una comparación en los tiempos de transmisión, la gran utilidad que tienen los algoritmos de compresión en los enlaces de alta velocidad.

- Con la utilización de esta aplicación, se puede ver una taza alta de compresión de datos tanto en archivos de gran tamaño como en los archivos pequeños. Con esto se demuestra lo importante que es utilizar algoritmos de compresión en redes de comunicación, en especial en las redes de transporte.

6.2 RECOMENDACIONES

- Se recomienda tener un mínimo de conocimiento de los fundamentos teóricos de estas tecnologías debido a que por desconocimiento del tema, al inicio la aplicación puede tornarse poco atrayente para las personas que utilicen el software por primera vez.

- Se recomienda leer el manual de usuario para resolver cualquier inquietud que surja por parte del usuario. Adicional, se recomienda seguir los pasos guías que la aplicación brinda como las etiquetas en cada una de las interfaces para una mejor manipulación del software.

- Se recomienda la instalación de la aplicación sobre un computador donde se disponga instalado un programa reproductor del formato Flash, el cual está desarrollado las animaciones. De igual manera el computador deberá tener instalado Microsoft Access ya que es en una base de datos donde se guarda los resultados estadísticos de los cálculos realizados por el algoritmo de compresión.

- La aplicación requiere de un mínimo de espacio en disco, razón por la cual se puede instalar en todo tipo de computador bajo los sistemas operativos Windows XP y Windows Vista ya que en estas versiones se han realizado las pruebas de funcionalidad.

- En cuestiones académicas, se recomienda la utilización de esta aplicación para la introducción a las tecnologías de transporte PDH y SDH. Con el complemento de esta herramienta se podrá comprender el accionar básico de las tecnologías de transmisión como PDH y SDH, debido a que estas

redes siguen siendo pilares en el campo de las comunicaciones se considera meritorio su estudio.

- Al manipular la aplicación de simulación, a pesar de que existen animaciones didácticas como complemento, se recomienda comenzar a examinar la parte de PDH ya que es la base para comprender SDH, adicional el formato de la trama PDH es más sencilla de entender. Una vez comprendida las ventajas de utilizar compresión sobre enlaces PDH seguir con la parte de SDH.

- Finalmente, se recomienda fomentar este tipo de trabajos, pues con esto se contribuye a un cambio significativo en la manera de aprendizaje. Se espera que este trabajo sirva como punto inicial para que se realicen aplicaciones orientadas al campo académico y así contar con ayudas didácticas acordes a tiempos actuales, relegando un poco el método tradicional de consulta.

REFERENCIAS BIBLIOGRAFICAS

1. BYEONG, Gi Lee, *Redes Integradas de Banda Ancha*, Edición, Artech House, 2002

2. HUIDROBO, José Manuel, *Tecnologías avanzadas de Telecomunicaciones*, Edición, Editorial EDIGAR, Buenos Aires Argentina, 2000

3. GARCIA, Tomas, *Alta velocidad y calidad de servicio en redes IP*, Edición, RA-MA Editorial, 2002

4. FLOOD, J. E, *Redes de Telecomunicaciones*, Edición, Second, 1997

5. Disponible en Internet: http://www.com.uvigo.es/~jfraile/asig_CO/PDH-SDH.pdf

6. Disponible en Internet: http://users.rcn.com/wpacino/technote.htm

7. Disponible en Internet: http://www.protocols.com/pbook/sonet.htm

8. Disponible en Internet: http://www.xelic-inc.com/Networking_Cores/XC_Cores/SONET_SDH/xcs3c.htm

9. Disponible en Internet: http://www.sincompromisos.com/Documentos/PDH-SDH/Estructura_de_trama_STM1.pdf

10. Disponible en Internet: http://www.trendtest.com/trendweb/resource.nsf/vlFileURLLookup/en%5E%5ESDH+monitoring/$FILE/SDH+monitoring+with+Victoria.pdf

GLOSARIO DE TERMINOS

ARCNET.- (Attached Resource Computing Network), Red de computación de recursos conectados, funciona con paso de testigo, es una red de banda base.

ARPANET.- Red creada por la Secretaría de Defensa de los Estados Unidos, actualmente es la red WAN mundial, es decir la Internet

ASCII.- (American Standard Code for Information Interchange), Código de caracteres basado en el alfabeto latino, utiliza 7 bits para representar los caracteres más 1 bit de paridad.

ASÍNCRONO- Que no tiene un intervalo de tiempo constante entre cada evento.

BAUDIOS.- Unidad informática utilizada para cuantificar el número de cambios de estado, o eventos de señalización, que se producen cada segundo durante la transferencia de datos.

BER.- (Bit Error Ratio), número de bits o bloques incorrectamente recibidos.

BROADCAST.- Dirección que designa al número general de una subred, donde todos los nodos de esa subred reciben la misma señal

CABECERA.- Información que se sitúa delante de los datos (por lo general en una transmisión) y que hace referencia a diferentes aspectos de la trama.

CAPA.- Cada una de los elementos que conforman una estructura jerárquica.

CIRCUITOS VIRTUALES.- Circuitos lógicos creados para asegurar una comunicación confiable entre dos dispositivos de la red.

CODIFICADOR.- Un codificador es un dispositivo lógico que recibe información por su entrada y la traduce a un código, el cual depende del tipo de codificador

CODIFICACIÓN HUFFMAN.- Este algoritmo consiste en la creación de un árbol binario, construido de tal forma que siguiendo desde la raíz a cada una de sus hojas se obtiene el código Huffman asociado a un símbolo dado.

CRC.- (Cyclical Redundancy Ckeching), Código de Redundancia Cíclica. Método matemático en el cuál se detectan errores en la información

DATAGRAMA.- Conjunto de estructurado de bytes que forma la unidad básica de comunicación del protocolo IP (en todas sus versiones).

DECNet.- (Digital Equipment Corporation), conocida como DECnet, arquitectura de red compuesta por cinco capas.

DNA.- (Distributed Network Arquitecture), arquitectura de red distribuida.

DQDB.- Estándar de red que equivale a la norma IEEE 802.6, consiste en buses unidireccionales, los cuales se conectan a todas las computadoras.

DEMODULADOR.- Técnicas utilizadas para recuperar la información transportada por una onda portadora.

DEMULTIPLEXOR.- Dispositivo que recibe a través de un medio de transmisión compartido una señal compleja multiplexada, separa las distintas señales y las encamina a su salida correspondiente.

DIAFONÍA.- Perturbación, cuando parte de los datos presentes en una señal aparece en otras señales.

DNA.- (Distributed Network Arquitecture), arquitectura de red distribuida,

DQDB.- Estándar de red que equivale a la norma IEEE 802.6, consiste en buses unidireccionales, los cuales se conectan a todas las computadoras.

EEPROM.- (Electrically-Erasable Programmable Read-Only Memory). Memoria ROM que puede ser programado, borrado y vuelto a programas eléctricamente.

ENCAPSULAMIENTO.- Sistema basado en colocar una estructura dentro de otra formando capas.

ENLACE.- Medio de comunicación sobre el que los nodos pueden comunicarse en la capa de enlace, es decir, la capa inmediatamente encima de IP.

ESTACIONES REMOTAS.- Equipo de una red que se encuentra ubicada fuera de la localía, pero que se gestiona desde una máquina en el sitio local.

ESTANDAR.- Especificación o norma que regula la realización de ciertos procesos o la fabricación de ciertos componentes para garantizar la interoperabilidad entre ellos.

ETD.- Equipo terminal de datos.

ETHERNET.- El método más usado de acceso a una LAN (Estándar IEE 802.3)

ETSI.- (European Telecomunications Standards Institute) organización de estandarización de la industria de las telecomunicaciones.

FDM.- (Frecuency Division Multiplexing) Multiplexación por División de Frecuencia.

FRECUENCIA.- Medida que indica el número de repeticiones de cualquier evento, luego estas repeticiones se dividen por el tiempo transcurrido.

GIF.- (Graphics Interchange Format), formato gráfico utilizado ampliamente en la web. GIF es un formato de codificación sin pérdida de calidad para imágenes con hasta 256 colores.

HALF-DUPLEX.- Método de envío de información bidireccional pero no simultáneo.

H.263.- Estándar de video comúnmente utilizado en videoconferencias.

H.323.- Recomendación ITU-T, define estándares para proveer sesiones de comunicación audiovisual sobre paquetes de red. Utilizado comúnmente para Voz sobre IP.

HDLC.- (Host Data Link Control), protocolo que define el fraccionamiento de los datos para la transmisión.

INTERFACE.- Lo que acopla un nodo a un enlace.

INTERNET.- Red de redes a escala mundial de millones de computadoras interconectadas con el conjunto de protocolos TCP/IP

IP.- (INTERNET PROTOCOL): Protocolo no fiable y sin conexión en el que se basa la comunicación por INTERNET. Su unidad es el datagrama.

JBIG.- (Joint Bi-level Image Experts Group), es un grupo de expertos que elabora normas para codificación de imágenes.

JERARQUÍA.- Orden de los elementos de una serie según su valor.

JERARQUÍA DIGITAL PLESIÓCRONA.- (PDH) Tecnología usada en telecomunicaciones, permite enviar varios canales telefónicos sobre un mismo medio de transmisión.

JPEG.- (Joint Photographic Experts Group), es un estándar diseñado para comprimir imágenes con 24 bits de profundidad (escala de grises). Solo trata imágenes fijas.

LAN.- (Local Area Network): Red local. Es la encargada de conectar computadores en distancias inferiores a 1Km.

LAP-D.- (Link Access Protocol for D-channel) Protocolo de control de enlace de datos para los canales tipo D que son usados para transporte de información de control y señalización y que nunca se separan de los canales B que transportan datos del usuario.

LEY A.- La Ley A es un sistema de cuantificación de señales de audio, usado para comprimir aplicaciones de voz humana. La Ley A se utiliza para sistemas PCM Europeos.

LEY µ.- La Ley µ es un sistema de cuantificación de señales de audio, usado para comprimir aplicaciones de voz humana. La Ley µ se utiliza para sistemas PCM Americanos.

MAU.- (Media Access Unit) Dispositivo multi-pórticos del equipamiento en el que se conectan hasta 16 estaciones de trabajo. La MAU brinda un control centralizado de las conexiones en red.

M-JPEG.- (Motion JPEG), formato de codificación para imágenes en movimiento.

MODULACIÓN.- Técnicas para transportar información sobre ondas de portadora, típicamente de onda sinusoidal.

MPEG.- (Moving Picture Experts Group), grupo de trabajo encargado de desarrollar estándares de codificación de audio y video, utilizando códecs de compresión de bajas pérdidas de datos.

MULTIPLEXACIÓN.- Combinación de dos o más canales de comunicación sobre un mismo medio físico usando un dispositivo llamado multiplexor.

MULTIPLEXOR.- Dispositivo que admite múltiples entradas y las reúne para transmitirlas juntas por un medio de transmisión.

OSI.- (Open Systems Interconnection) Interconexión de Sistemas Abiertos. Modelo de comunicaciones estándar entre los diferentes terminales y host. Las comunicaciones siguen unas pautas de siete niveles preestablecidos que son Físico, Enlace, Red, Transporte, Sesión, Presentación y Aplicación.

OVERHEAD.- Pérdida de rendimiento.

PAQUETE.- Es el encabezado del IP más la carga.

PCM.- (Pulse Code Modulation), es una representación digital de una señal analógica.

POOLING.- Técnica de sondeo para redes tipo LAN.

PORTADORA.- Sistema genérico de telecomunicaciones para los sistemas digitales multiplexados.

PROTOCOLOS.- Conjunto de reglas que establece cómo debe realizarse una comunicación.

SINCRONIZACIÓN.- Realización simultánea de dos procesos.

SÍNCRONO.- Que tiene un intervalo de tiempo constante entre cada evento. Son procesos síncronos los que dependen de un acontecimiento exterior que los acciona.

SNA.- (System Network Arquitecture), Arquitectura de redes y sistemas

SDM.- (Space Division Multiplexing) Multiplexación por División en el Tiempo.

TDM.- (Time Division Multiplexing) Multiplexación por División de Tiempo.

TCP.- (Transmission Control Protocol): Protocolo de nivel superior que permite una conexión fiable y orientada a conexión mediante el protocolo IP.

TRANSMISIÓN.- Transferencia de datos a través de un canal de comunicación.

UDP (User Datagram Protocol): Protocolo no fiable y sin conexión basado en el Protocolo IP.

VLAN.- Virtual LAN, redes LAN virtuales. Una VLAN segmenta lógicamente la red mediante el uso de switches.

VoIP.- (Voz Over IP), voz sobre el protocolo IP, grupo de recursos que hacen posible que la señal de voz viaje sobre la Internet empleando el protocolo IP.

WAN.- (Wide Area Network) Red de comunicación de datos que tiene una extensa covertura,

WAN.- (Wireless LAN) Redes LAN Inalámbricas, sistema de comunicación de datos inalámbrico flexible, muy utilizado como alternativa a las tradicionales redes LAN. Utiliza tecnología de radiofrecuencia que permite mayor movilidad a los usuarios.

WEP.- (Wired Equivalent Privacy) Sistema de encriptación, utilizado en redes wireless por cuestión de seguridad.

X.25.- Estándar para redes de paquetes recomendado por CCITT. X.25 es una norma de interfaz para las redes de paquetes de gran cobertura. Ofrece un servicio orientado a conexión, es fiable, y ofrece multiplexación

ANEXOS

ANEXO A

Anexo A. Código Fuente.

CLASE: HUFFMAN

```
class Huffman
    {
        public string Control; //verifica la formación del árbol binario
        long[] ArrayPesos; //vector de 256 posiciones que almacena los caracteres del archivo a comprimir
        RamaNodo[] Ramas; //vector que almacena el árbol binario
        int NumChr; //variable que almacena el valor entero correspondiente al códogo ASCII del caracter leido
        long PesoTotal; //variable que almacena el tamaño del Archivo que se va a comprimir
        string NombreArchivo; //variable que almacena el nombre del archivo que se va a comprimir
        public Huffman() {//constructor de la clase Huffman
            ArrayPesos = new long[256]; //vector de 256 para almacenar los valores leidos del archivo
        }

        private void Orderar(int Limit)
        {
            RamaNodo RTemp; // variable que realiza el cambio de posición de una hoja a rama
            RTemp = new RamaNodo();//instancia de RamaNodo
            if (Limit > Ramas.Length)
            {
                Limit = Ramas.Length; //asigna a Limit el valor del numero de posiciones del vector Ramas[]
            }
            for (int i = 0; i < Limit;i++ )//contador para ordenar el nuevo vector Ramas
            {
                for (int j = i; j < Limit; j++)//bucle para obtener nivel y peso de cada posicion del vector
                {
                    if ((Ramas[i].GetNivel() *Limit + Ramas[i].GetPeso()) > (Ramas[j].GetNivel()*Limit + Ramas[j].GetPeso()))//verifica si el valor de las ramas es alto (mayor repeticiones)
                    {
                        RTemp = Ramas[i];//Asigna en RTemp el valor del vector con mayor frecuencia
                        Ramas[i] = Ramas[j]; //ordena el árbol binario
                        Ramas[j] = RTemp; //actualiza RTemp
                    }
                }
            }
        }

        public long TamañoArchivo(){
            return PesoTotal; //retorna el peso del archivo seleccionado
        }

        public int NumeroCaracteres(){
            return NumChr; //retorna el número ASCII del caracter leido
        }
```

```
        public long CalcularArchivoPesos(string Archivo)
          { //recibe el nombre del archivo de la lista
              PesoTotal = 0; //variable en 0
              NumChr = 0; //variable en 0
              NombreArchivo = Archivo; //asigna el nombre del archivo
              for (int i = 0; i < 256; i++)//for para inicializar un vector
              {// de 256 posiciones
                  ArrayPesos[i] = 0; //a cada posicion se le asigna 0
              }
              if ( System.IO.File.Exists(Archivo)) // si el archivo existe
              {
                  try
                  {
                      int s = 0; //variable s inicializada en 0
                      FileStream stream = new FileStream(NombreArchivo,
FileMode.Open,  FileAccess.Read); //asigna a stream el nombre del archivo y da
permisos de lectura y escritura
                     PesoTotal  =  new  FileInfo(NombreArchivo).Length;
                    //obtiene el tamaño del archivo
                      BinaryReader reader = new BinaryReader(stream);
//reader para lectura del archivo en forma binaria

                      for (long i=0;i<PesoTotal;i++)//bucle para leer el archivo
                      {
                          s = reader.ReadByte();//variable s almacena el valor del
caracter leido
                          if (ArrayPesos[s] == 0) //verifica si la posicion del
[] es 0
                          {
                                NumChr++;
                          }//incrementa el contador de caracteres
                                ArrayPesos[s]++;//incrementa  el  contador  de
caracteres para la posicion de []
                      }
                      reader.Close();//cierra el lector
                      stream.Close();//cierra el stream
                  }
                  catch
                  {
                      NumChr = 0; //asigna 0 a NumChr
                      PesoTotal = 0; //asigna 0 a PesoTotal
                      return -1; //retorna -1
                  }
              }
              return PesoTotal; //retorna PesoTotal calculado
          }

          public int CrearArbol()
          {
              if (NumChr > 0)
              {
                  Ramas = new RamaNodo[(NumChr * 2) - 1]; //crea el vector
Ramas tipo RamaNodo
                  int RamaAc = 0; //inicializa la variable RamaAc = 0
                  for (int i = 0; i < 256; i++)//bucle con contador de 256
                  {
                      if (ArrayPesos[i] > 0)
                      {
                          Ramas[RamaAc]  =  new  RamaNodo(ArrayPesos[i],
i);//instancia de RamaNodo.cs
                          RamaAc++;//incrementa la posición del vector Ramas[]
```

```
                    }
                }
                while (RamaAc < Ramas.Length)                {
                    Orderar(RamaAc); //llama a Orderar() pasando el valor de
RamaAc (rama actual)
                    Ramas[RamaAc] = new RamaNodo(Ramas[0], Ramas[1]);
//llama a RamaNodo y pasa un valor de 0 y 1 (Rama I, Rama D)
                    RamaAc++;//incrementa el numero de RamaAc
                }
                SubirHojas();//llama a SuborHojas()
                return Ramas.Length; //retorna la longitud del vector Ramas
            }
            else
            {
                return (-1); //retorna -1
            }
        }

        private int SubirHojas()
        {
            int x, y, r; //declara variables locales
            r = 0; //inicializa las variables locales
            RamaNodo Rtemp = new RamaNodo();//instancia de RamaNodo
            x = 0; y = 0;
            if (Ramas.Length > 0)
            {
                while (y < Ramas.Length) //mientras y < Ramas.Length
                {
                    if (Ramas[y].GetEsRama())//si Ramas[y] es rama
                    {
                        y++;//incrementa y
                    }
                    else
                    {
                        Rtemp = Ramas[x]; //almacena en variable Rtemp
                        Ramas[x] = Ramas[y]; //se cambian los valores de las
posiciones del árbol binario
                        Ramas[y] = Rtemp; //actualiza la variable RTemp
                        x++;//aumenta x
                        y++;//aumenta y
                    }
                }
            }

            if (Ramas[NumChr].GetEsRama())//si la posicion del vector es Rama
            {
                r = 2; //almacena 2 en la variable r
            }
            else //caso contrario
            {
                r = -1; //almacena -1 en la variable r
            }
            Control = "Subir hojas " + r.ToString();//controla el nivel del
árbol binario
            return r; //retorna el valor de r
        }

        public int PesoTabla()
        {
            int vPeso = 0; //inicializa la variable vPeso
```

```
            for (int i = 0; i < NumChr; i++)//bucle con contador para trabajar
con el número de Caracteres
            {
                vPeso = vPeso + 1 + Ramas[i].GetCodigo().Length; //suma el
peso de los codigos
            }
            return vPeso; //retorna el peso obtenido
        }

        public int PesoZip()
        {
            int vPeso=0; //inicializa la variable vPeso

            if (System.IO.File.Exists(NombreArchivo)) //verifica que exista el
archivo seleccionado
            {
                try
                {
                    int s = 0; //inicializa variables
                    int p = 0; //inicializa variables
                    FileStream stream = new FileStream(NombreArchivo,
FileMode.Open, FileAccess.Read); //lanza un lector con permisos de acceso y
escritura para el archivo seleccionado
                    BinaryReader reader = new BinaryReader(stream);
//lanza un stream para lectura
                    for (long i = 0; i < PesoTotal; i++)//bucle hasta que sea
menor que el peso del archivo
                    {
                        s = reader.ReadByte();//lector almacena en s los datos
                        bool Seguir = true; // variable asignada a true
                        while (Seguir) //mientras  Seguir = true
                        {
                            if (Ramas[p].GetChr() == s) //compara si el valor
del vector Ramas[] s = al leido
                            {
                                vPeso = vPeso +
Ramas[p].GetCodigo().Length; //calcula el peso de los códigos
                                Seguir = false; //asigna falso
                                p = -1; //asigna -1 a p
                            }
                            p++;//incrementa p
                        }
                    }
                    reader.Close();//cierra el lector
                    stream.Close();//cierra el stream
                }
                catch
                {
                    NumChr = 0; //asigna 0 a NumChr
                    PesoTotal = 0; //asigna 0 a PesoTotal
                    return -1; //retorna -1
                }
            }
            vPeso = (vPeso / 8) + 1; //divide el valor de vPeso para 8 y los
redondea para obtener un valor en bytes
            return vPeso; //retorna vPeso
        }
    }
```

CLASE: FRMVARIOSE1V2

```
public partial class FrmVariosE1V2 : Form
    {
        public FrmVariosE1V2(string Medio)//recibe el valor que envia MDIParent
        {
            InitializeComponent(Medio);//llama a FrmVariosE1V2.designer e inicializa los componentes
        }


        private void FrmVariosE1V2_Load(object sender, EventArgs e)
        {
            medios1.FijarUsuarios(ref ctnUsuarios1);//instancia para fijar las variables de Usuarios
            ctnUsuarios1.FijarMedios(ref  medios1);//instancia para fijar las variables de Medios
        }


    }
```

CLASE: CTNUSUARIOS

```
public partial class CtnUsuarios : UserControl
    {
        private string FileFlash = ""; //variable para ubicar la ruta de la animación
        public Medios VGMedios; //variable global de Medios, contiene variables de usuario
        public int UsuariosCtn = 0;//variable para controlar los usuarios
        public Usuario001[] ArraryUsuarios; //vector de tipo Usuario001 para almacenar datos de los usuarios

        public void FijarMedios(ref Medios vlMedios)
        {
            VGMedios = vlMedios; //asigna a VGUsuarios el valor del medio pasado
        }

        public CtnUsuarios()
        {
            InitializeComponent();//inicializa los componentes de FRM CtnUsuarios
            ArraryUsuarios = new Usuario001[200]; //crea un vector de 200 posiciones para almacenar usuarios
        }

        private void AddUsuario()
        {
            ArraryUsuarios[UsuariosCtn] = new Usuario001();//inicializa la variable tipo vector ArraryUsuarios como instancia de Usuario001
            ArraryUsuarios[UsuariosCtn].label1.Text = "Usuario " + UsuariosCtn.ToString();//asigna el número de Usuario
            flowLayoutPanel1.Controls.Add(ArraryUsuarios[UsuariosCtn]);//agrega al layout los controles de usuario
            ArraryUsuarios[UsuariosCtn].FijarSubMedio(ref VGMedios); //llama a FijarSubMedio() en Usuario001 para definir si el medio es V1 o V2
            UsuariosCtn++;//incrementa el numero de usuarios
        }

        private void DelUsuario()//borra usuarios
        {
            UsuariosCtn--;//disminuye el contador del número de usuarios
            for (int i = 0; i < ArraryUsuarios[UsuariosCtn].listBox1.Items.Count; i++)//verifica que exista usuarios y si los hay los elimina
            {

ArraryUsuarios[UsuariosCtn].MyMedio.Liberar(ArraryUsuarios[UsuariosCtn].listBox1.Items[i].ToString());//llama a liberar en Medios para eliminar un canal de la lista
            }
            ArraryUsuarios[UsuariosCtn].Dispose();//actualiza el vector de usuarios
        }

        private void numericUpDown1_ValueChanged(object sender, EventArgs e)
        {
             string appPath = Path.GetDirectoryName(Application.ExecutablePath); //asigna a la variable la ubicación donde se ejecuta la animación
            switch (this.ParentForm.Text.ToString())//selección de un caso
            {
                case "E1"://Si se selecciona un E1 se ejecuta la animación E1
```

```
                    FileFlash = appPath + "\\Simul_E1.swf";
                    break;
                case "AU3":// Si se selecciona un AU3 se ejecuta la animación VC3
                    FileFlash = appPath + "\\Simul_VC3.swf";
                    break;
                case "AU4":// Si se selecciona un AU4 se ejecuta la animación VC4
                    FileFlash = appPath + "\\Simul_VC4.swf";
                    break;
            }
            if (axShockwaveFlash1.Movie != FileFlash)
            {
                axShockwaveFlash1.Movie = FileFlash; //asigna al control Flash
la ruta de la animación seleccionada
                axShockwaveFlash1.Play();//ejecuta la animación seleccionada
            }

            if (numericUpDown1.Value > UsuariosCtn)

            {
                AddUsuario();//llama a la función AddUsuario()
            }
            else
            {
                DelUsuario();//llama a la función DelUsuario
            }
            ActualizarCanales();//llama a la función ActualizarCanales
        }

        public void ActualizarCanales()
        {
            for (int i = 0; i < UsuariosCtn; i++)
            {

                ArraryUsuarios[i].listBox1.Items.Clear();//limpia  los  items
del listbox1
                ArraryUsuarios[i].listBox2.Items.Clear();//limpia  los  items
del listbox2

                for (int j = 0; j < VGMedios.MediosCtn; j++)//contador para
# de trama
                {

                for       (int       k       =       0;       k       <
      VGMedios.ArrayMedios[j].Canales.Length; k++)//contador para # de canal
                    {

                        if (VGMedios.ArrayMedios[j].Canales[k].Libre
                        { )//verifica si el canal elegido para el AB esta libre para
mostrarlo en el listBox de canales disponibles
string
tipo=VGMedios.ArrayMedios[j].Canales[k].ToString().Substring(0,2);//obtiene
el valor del Medio
Boolean filtrar = (((tipo == "E1") && (k == 0)) || ((tipo == "E1") && (k
== 16)));//filtra Canales 0 y 16 para no utilizarlos

                            if (!filtrar)
                            {
ArraryUsuarios[i].listBox2.Items.Add(VGMedios.ArrayMedios[j].Canales[k].T
oString());//agrega el canal disponible en el listBox de canales disponibles
                        }
```

```
                        else
                        {

                            if
(VGMedios.ArrayMedios[j].Canales[k].UsuarioCanal == i)
                            {

ArraryUsuarios[i].listBox1.Items.Add(VGMedios.ArrayMedios[j].Canales[k].T
oString());//al canal ocupado
                            }//se lo muestra en el otro listBox (listado de canales
seleccionados) no en el listado
                        }// de canales disponibles
                    }
                }
            }
        }
    }
```

CLASE: MEDIOS

```
public partial class Medios : UserControl
    {
        public CtnUsuarios VGUsuarios; //variable global para usuarios y medios
        public MedioV2[] ArrayMedios; //vector de tipo MedioV2
        public int MediosCtn=0; //variable inicializada en 0
        private string MedioTipo; //almacena el tipo de tecnología de TX elegida

        public void Reser(string Canal, string Usuario, int IntUser)
        {
            string[] coordenadas; //inicializa la variable coordenadas
            coordenadas = Canal.Substring(Canal.IndexOf("(") + 1,
Canal.Substring(Canal.IndexOf("(") + 1).Length - 1).Split(','); //
descompone las coordenadas del número de canal elegido y los almacena en el vector
coordenadas[]
ArrayMedios[Convert.ToInt32(coordenadas[0])].Canales[Convert.ToInt32(coor
denadas[1])].textBox1.Text=Usuario; //coloca el nombre de usuario en el textBox1
ArrayMedios[Convert.ToInt32(coordenadas[0])].Canales[Convert.ToInt32(coor
denadas[1])].textBox1.BackColor = MiColor; //asigna un color al textBox1

ArrayMedios[Convert.ToInt32(coordenadas[0])].Canales[Convert.ToInt32(coor
denadas[1])].UsuarioCanal = IntUser; //asigna un usuario al canal de la trama
ArrayMedios[Convert.ToInt32(coordenadas[0])].Canales[Convert.ToInt32(coor
denadas[1])].Libre = false; //asigna OCUPADO al número de canal elegido para el AB
VGUsuarios.ActualizarCanales();//llama a ActualizarCanales
        }

        public void Liberar(string Canal)
        {
            string[] coordenadas; //vector para almacenar las coordenadas del canal
            coordenadas = Canal.Substring(Canal.IndexOf("(") + 1,
Canal.Substring(Canal.IndexOf("(") + 1).Length - 1).Split(','); //captura
en el vector coordenadas las posiciones del canal utilizado

ArrayMedios[Convert.ToInt32(coordenadas[0])].Canales[Convert.ToInt32(coor
denadas[1])].textBox1.Text="00000000";//libera el canal del usuario seleccionado
ArrayMedios[Convert.ToInt32(coordenadas[0])].Canales[Convert.ToInt32(coor
denadas[1])].textBox1.BackColor = Color.White; //asigna el color Blanco al canal
liberado
ArrayMedios[Convert.ToInt32(coordenadas[0])].Canales[Convert.ToInt32(coor
denadas[1])].UsuarioCanal = -1; //disminuye el límite de canales
ArrayMedios[Convert.ToInt32(coordenadas[0])].Canales[Convert.ToInt32(coor
denadas[1])].Libre = true; //asigna libre al canal
            VGUsuarios.ActualizarCanales();//llama a actualizarCanales()
        }

        public void FijarUsuarios(ref CtnUsuarios Usuarios)
        {
            VGUsuarios=Usuarios; //asigna a VGUsuarios las variables que incluye en
cada perfil de usuario
        }

        public Medios(string Medio)
        {
            InitializeComponent();//inicializa los componentes de Medios
            MedioTipo = Medio; //recibe el tipo de variable de TX seleccionado para
la simulación
            ArrayMedios = new MedioV2[200]; //crea un vector de 200 posiciones
para almacenar los usuarios
        }
```

```
        private void AddMedio()//Agrega una trama
        {
            ArrayMedios[MediosCtn] = new MedioV2(MedioTipo, MediosCtn);
//se instancia a MedioV2
            ArrayMedios[MediosCtn].label1.Text=MedioTipo + " (" +
MediosCtn.ToString() + ")";//asiga el valor del numero de trama
            flowLayoutPanel1.Controls.Add(ArrayMedios[MediosCtn]); //agrega
medios al FlowLayout
            MediosCtn++;//incrementa en 1 la variable MedioSCtn
        }

        private void DelMedio()//elimina tramas
        {

ArrayMedios[Convert.ToInt32(numericUpDown1.Value)].Dispose();//oculta el
vector ArrayMedios
            MediosCtn--;//desminuye a 1 el contador de la variable MediosCtn
        }

        private void numericUpDown1_ValueChanged(object sender, EventArgs
e)
        {
            if (numericUpDown1.Value > MediosCtn)
            {
                AddMedio();//llama a AddMedio
            }
            else
            {
                DelMedio();//llama a DelMedio
            }
            VGUsuarios.ActualizarCanales();//llama a ActualizarCanales
        }
    }
```

CLASE: MEDIOV2

```
public partial class MedioV2 : UserControl
    {
        public Canal[] Canales; //variable de tipo Canal

        public MedioV2()//constructor de MedioV2
        {
            InitializeComponent();//inicializa los componentes
        }

        public MedioV2(string Tipo, int NumMedio)
        {
            InitializeComponent();//iniciaiza los componentes
            switch (Tipo) //recibe el tipo de trama a utilizarse en la simulación
            {
                case "E1":
                    Canales = new Canal[32]; //crea un vector de 32 posiciones
                    for (int i = 0; i < 32; i++)//bucle con contador de 32
posiciones
                    {
                        Canales[i] = new Canal();//se crea una instancia de
Canal
                        Canales[i].FijarDatos(Tipo, NumMedio, i); //pasa
los valores de tipo de trama, número de trama y número de canal a FijarDatos
                        Canales[i].label1.Text = "Canal " +
i.ToString();//asigna a nombres a los Canales
                        Canales[i].textBox1.Text = "00000000";//asigna a
cada canal el valor de 00000000
                        flowLayoutPanel1.Controls.Add(Canales[i]); //añade
el canal al flowLayout el Canal
                    }
                    break;
                case "AU3":
                    Canales = new Canal[3]; //define un vector de 3 posiciones
                    for (int i = 0; i < 3; i++)
                    {
                        Canales[i] = new Canal();();//se crea una instancia de
Canal
                        Canales[i].FijarDatos(Tipo, NumMedio, i); //pasa
los valores de tipo de trama, número de trama y número de canal a FijarDatos
                        Canales[i].label1.Text = "Canal " +
i.ToString();//asigna al label como Canalx
                        Canales[i].textBox1.Text = "00000000";//asigna a
cada canal el valor de 00000000
                        flowLayoutPanel1.Controls.Add(Canales[i]); //añade
el canal a flowlayout
                    }
                    break;
                case "AU4":
                    Canales = new Canal[1]; //define un vector de 1 posición
                    for (int i = 0; i < 1; i++)
                    {
                        Canales[i] = new Canal();();();//se crea una
instancia de Canal
                        Canales[i].FijarDatos(Tipo, NumMedio, i); //pasa
los valores de tipo de trama, número de trama y número de canal a FijarDatos
                        Canales[i].label1.Text = "Canal " +
i.ToString();//asigna al label como canal
                        Canales[i].textBox1.Text = "00000000";//asigna a
cada canal el valor de 00000000
```

```
                    flowLayoutPanel1.Controls.Add(Canales[i]);  //añade
el canal al flowlayout
                    break;
                }
            }

        }
```

CLASE: CANAL

```
public partial class Canal : UserControl
    {
        public Boolean Libre = true; //variable para canal libre o ocupado
        private int numMedio; //variable para el numero de tramas
        public int UsuarioCanal; //variable para el canal del usuario
        private int numCanal; //variable para el numero de canal
        private string NomMedio; //variable para el nombre del medio

        public void FijarDatos(string nNombre, int nMedio, int nCanal)
        {
            NomMedio = nNombre; //almacena el tipo de trama
            numMedio = nMedio; //almacena el dato de número de trama
            numCanal = nCanal; //almacena el dato del time slot
        }

        public override string ToString()
        {
            string vl; // inicializa la variable vl
            vl = NomMedio + "(" + numMedio + " , " + numCanal +
")";//asigna a vl el nombre, número de la trama y número de time slot
            return vl; //retorna null
        }

        public Canal()
        {
            InitializeComponent();//inicializa los componentes
        }

    }
```

CLASE: USUARIO001

```
public partial class Usuario001 : UserControl
    {
        private Int32 Velocidad = 0; //variable para manejo de velocidades de tx
        public Medios MyMedio; //variable de tipo Medios
        public CtnTU12 MyMedioTU12; //variable de tipo de contenedor CtnTU12
        Huffman ArchTemp; //variable de tipo Huffman
        LimitarCanales[] limites; //variable de tipo LimitarCabnales
        int NumMax = 0; //variable para control de anchos de banda

        private bool ArchivoExiste(string Archivo)
        {
            return true; //retorna verdadero si el archivo existe
        }

private void QuitarUno()
        {
                if (MyMedio != null)
                {

MyMedio.Liberar(listBox1.Items[0].ToString());//instancia al método Liberar
justamente para liberar un canal que no se desee utilizar
                }
                else
                {
                    int S, x, y, z;
                    S =
Convert.ToInt32(listBox1.Items[0].ToString().Substring(5, 2)); //convierte a
entero el indicador del número de trama
                    x =
Convert.ToInt32(listBox1.Items[0].ToString().Substring(10, 1)); //convierte
a entero el indicador de la posición x del vector de canales TU-12
                    y =
Convert.ToInt32(listBox1.Items[0].ToString().Substring(11, 1)); //convierte
a entero el indicador de la posición y del vector de canales TU-12
                    z =
Convert.ToInt32(listBox1.Items[0].ToString().Substring(12, 1)); //convierte
aentero el indicador de la posición z del vector de canales TU-12
M yMedioTU12.LiberarCanal(S, x, y, z); //libera el canal de la trama TU-12 con las
coordenadas anteriormente procesadas
                }
        }

        public void SubirLimites(String TipoCanal)
        {
            switch (TipoCanal)
            {
                case "E1":
                    limites= new LimitarCanales[6]; //crea un vector de 6
posiciones para l0s 6 posibles anchos de banda que se puede seleccionar en esta opción
                    limites[0] = new LimitarCanales("  64kb",1); //asigna a
la posicion 1 64
                    limites[1] = new LimitarCanales(" 128kb",2); //asigna a
la posicion 2 128
                    limites[2] = new LimitarCanales(" 256kb",4); //asigna a
la posicion 3 256
                    limites[3] = new LimitarCanales(" 512kb",8); //asigna a
la posicion 4 512
                    limites[4] = new LimitarCanales("1024kb",16); //asigna
a la posicion 5 1024
```

```
                    limites[5] = new LimitarCanales("2048kb",32); //asigna
a la posicion 6 2048
                    break;
                case "STM1_TU12"://limita el numero de posibles anchos de banda
para 63 canales TU12 de 2Mbps
                    limites = new LimitarCanales[63]; //limita un vector de 63
posiciones
                    for (int j = 0; j < 63; j++)
                    {
                        limites[j] = new LimitarCanales(((j + 1) *
2).ToString() + " Mbps", j + 1); //asigna el numero de canales a elegir según el AB
requerido
                    }
                    break;

                case "AU3"://limita el numero de posibles anchos de banda para 3
canales AU3 de 45Mbps
                    limites = new LimitarCanales[3]; //limita un vector de
tres posiciones
                    for (int j = 0; j < 3; j++)
                    {
                        limites [j]= new LimitarCanales(((j + 1) *
45).ToString() + " Mbps", j + 1); //asigna el numero de canales a elegir según el
AB requerido
                    }
                    break;

                case "AU4"://limita el numero de posibles anchos de banda para AU4
solo un AB de 140Mbps
                    limites = new LimitarCanales[1]; //limita un vector de una
posicion
                    for (int j = 0; j < 1; j++)
                    {
                        limites[j] = new LimitarCanales(((j + 1) *
140).ToString() + " Mbps", j + 1); //asigna un ancho de banda de 140Mbps
                    }
                    break;
            }
            if (limites != null)
            {
                for (int i = 0; i < limites.Length; i++)
                {
                    comboBox1.Items.Add(limites[i]); //asigna los posibles
valores de AB al ComboBox
                }
            }
        }

        public void LlenarMaximos(int Caso)
        {
            switch (Caso)
            {
                case 0:
                    comboBox1.Items.Add("  64kb");//añade al combobox 64
                    comboBox1.Items.Add(" 128kb");//añade al combobox 128
                    comboBox1.Items.Add(" 256kb");//añade al combobox 256
                    comboBox1.Items.Add(" 512kb");//añade al combobox 512
                    comboBox1.Items.Add("1024kb");//añade al combobox 1024
                    comboBox1.Items.Add("2048kb");//añade al combobox 2048
                    break;
            }
        }
```

```
        public void FijarSubMedio(ref Medios MedioOri)
        {
            MyMedio = MedioOri; //Asignación de valores a la variable MyMedio con
los parámetros que utiliza el usuario
        }

        public void FijarSubMedio(ref CtnTU12 MedioOri)
        {
            MyMedioTU12 = MedioOri; //Asignación de valores a la variable MyMedio
con los parámetros que utiliza el usuario
        }

        public Usuario001()
        {
            InitializeComponent();//inicializa los componentes
            ArchTemp = new Huffman();//instancia de Huffman
        }

        private void Usuario001_Load(object sender, EventArgs e)
        {
            SubirLimites(this.ParentForm.Text); //captura el valor del nombre
del Form abierto
        }

        private void listBox2_DoubleClick(object sender, EventArgs e)
        {
            if (listBox2.SelectedItems.Count > 0)
            {
                if (NumMax > listBox1.Items.Count)
                {
                    if (MyMedio != null)
                    {
                        MyMedio.Reser(listBox2.SelectedItem.ToString(),
label1.Text,  Convert.ToInt32(label1.Text.Substring(label1.Text.IndexOf("
"))));  //obtiene el indicador del número de usuario
                    }
                    else
                    {
                        int S, x, y, z;
                        S                                                =
Convert.ToInt32(listBox2.SelectedItem.ToString().Substring(5,         2));
//convierte a entero el indicador del número de trama
                        x                                                =
Convert.ToInt32(listBox2.SelectedItem.ToString().Substring(10,        1));
//convierte a entero el indicador de la posición x del vector de canales TU-12
                        y                                                =
Convert.ToInt32(listBox2.SelectedItem.ToString().Substring(11,        1));
//convierte a entero el indicador de la posición y del vector de canales TU-12
                        z                                                =
Convert.ToInt32(listBox2.SelectedItem.ToString().Substring(12,        1));
//convierte a entero el indicador de la posición z del vector de canales TU-12
MyMedioTU12.FijarCanal(S, x, y, z, label1.Text); //asigna el canal de la trama
TU-12 con las coordenadas anteriormente procesadas
                    }
                }
            }
        }

        private void listBox1_DoubleClick(object sender, EventArgs e)
        {
            if (listBox1.SelectedItems.Count > 0)
            {
```

```
                if (MyMedio != null)
                {
MyMedio.Liberar(listBox1.SelectedItem.ToString());//libera el canal
seleccionado
                }
                else
                {
                    int S, x, y, z;
                    S =
Convert.ToInt32(listBox1.SelectedItem.ToString().Substring(5, 2));
//convierte a entero el indicador del número de trama
                    x =
Convert.ToInt32(listBox1.SelectedItem.ToString().Substring(10, 1));
//convierte a entero el indicador de la posición x del vector de canales TU-12
                    y =
Convert.ToInt32(listBox1.SelectedItem.ToString().Substring(11, 1));
//convierte a entero el indicador de la posición y del vector de canales TU-12
                    z =
Convert.ToInt32(listBox1.SelectedItem.ToString().Substring(12, 1));
//convierte a entero el indicador de la posición z del vector de canales TU-12
MyMedioTU12.LiberarCanal(S, x, y, z); //libera el el canal de la trama TU-12 con
las coordenadas anteriormente procesadas
                }
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            OFD1.ShowDialog();/Muestra el OpenFileDialog1
            for (int i = 0; i < OFD1.FileNames.Length; i++)
            {
                if (ArchivoExiste(OFD1.FileNames[i])) //confirma la existencia
del archivo seleccionado
                {
listBox3.Items.Add(OFD1.FileNames[i]); //agrega a la lista el npmbre del archivo
seleccionado
                }
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (listBox3.SelectedItems.Count > 0)
            {
                listBox3.Items.Remove(listBox3.SelectedItem);  //èlimina  el
nombre del archivo del listbox
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            string appPath =
Path.GetDirectoryName(Application.ExecutablePath);/captura el Path del Archivo
            if (listBox3.Items.Count > 0)
            {
                // borrar datos
                string strConn = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=.\\DBEstadisticas.mdb"; //establece una ruta de conexión a la base de datos
                OleDbConnection myAccConection = null; //establece un objeto
de conexión a la base de datos
                string strCommand; //variable para realizar comandos de ejecución
```

```
                strCommand   =   "DELETE   FROM   TBL_Estadisticas   WHERE
Est_Usuario='" + label1.Text + "'"; //borra los datos de la tabla
                try
                {
                    myAccConection = new OleDbConnection(strConn); //crea
una conexión a la BD
                    myAccConection.Open();//abre la conexión a la BD

                    OleDbCommand myCommand = new OleDbCommand(strCommand,
myAccConection); //establece la conexión a la base de datos
                    myCommand.ExecuteNonQuery();//borra la base de datos
                    myAccConection.Close();//cierra la conexión a la DB
                }
                catch
                {
                }

                for (int i = 0; i < listBox3.Items.Count; i++)
                {
                    // definir variables

ArchTemp.CalcularArchivoPesos(listBox3.Items[i].ToString());/obtiene      el
tamaño del archivo
                   ArchTemp.CrearArbol();//llama a CrearArbol en Huffman.cs
                    string vl_Nombre, vl_Usuario, vl_Medio; //asignación de
variables
                    long vl_Peso; //asignación de variables
                    Int32 vl_zip, vl_tabla, vl_distancia, vl_AnchoBanda;
                    double vl_velocidad; //asigna variables
                    // Calcular valores
                    vl_Nombre  =  listBox3.Items[i].ToString();//almacena  el
nombre del archivo seleccionado
                    vl_Peso = ArchTemp.TamañoArchivo();//almacena el peso del
archivo
                    vl_zip = ArchTemp.PesoZip();//almacena el peso del archivo
comprimido
                    vl_tabla  =  ArchTemp.PesoTabla();//almacena  el  peso  de  la
tabla
                    vl_Usuario = label1.Text; //asigna el usuario al textbox
                    vl_Medio                                                =
this.ParentForm.Controls["comboBox2"].Text; //selecciona  el medio
                    vl_velocidad = 0; //setea la velocidad a 0
                    switch
(this.ParentForm.Controls["comboBox2"].Text.Trim())
                    {
                        case "Radio Frecuencia":
                            vl_velocidad  =  0.00000003; //asigna  el  valor  de
velocidad para Radio Frecuencia
                            break;
                        case "Cobre":
                            vl_velocidad  =  0.000000023; //asigna  el  valor  de
velocidad para el Cobre
                            break;
                        case "Fibra Optica":
                            vl_velocidad  =  0.00000002; //asigna  el  valor  de
velocidad la Fibra óptica
                            break;
                    }
                    vl_distancia                                            =
Convert.ToInt32(this.ParentForm.Controls["numericUpDown1"].Text);  //asigna
el valor de distancia del enlace
```

```
                    vl_AnchoBanda = listBox1.Items.Count * 64; //asigna el
valor de velocidad*64

                    strCommand = "INSERT INTO TBL_Estadisticas (
Est_nombre_arhivo, Est_peso, Est_Comprimido, Est_Tabla, Est_Usuario,
Est_Medio, Est_VelocidadT, Est_Distancia, Est_AnchoBanda ) values ('" +
vl_Nombre + "'," + vl_Peso + "," + vl_zip + "," + vl_tabla + ",'" +
vl_Usuario + "','" + vl_Medio + "'," + vl_velocidad + "," + vl_distancia
+ "," + vl_AnchoBanda + ")";//comando de inserción de datos en la base de datos
                    try
                    {
                        myAccConection.Open();//abre la base de datos
                        OleDbCommand myCommand = new
OleDbCommand(strCommand, myAccConection); //establece un comando de ejecucion
                        myCommand.ExecuteNonQuery();//ejecuta el comando
                        myAccConection.Close();//cierra la base de datos
                    }
                    catch
                    {
                    }
                }
            }


this.TBL_EstadisticasTableAdapter.FillByUsuario(this.DBEstadisticasDataSe
t.TBL_Estadisticas, label1.Text); //revisa los datos de la base de datos
            switch(Velocidad){
            case 64:
            reportViewer1.Visible = true; //muestra el reporte1
            reportViewer2.Visible = true; // muestra el reporte2
            reportViewer3.Visible = true; // muestra el reporte3
            this.reportViewer1.RefreshReport();//actualiza el reporte1
            this.reportViewer2.RefreshReport();//actualiza el reporte2
            this.reportViewer3.RefreshReport();//actualiza el reporte3
            break;
            case 2048: //velocidad = 2048
            reportViewer4.Visible = true; //muestra el reporte4
            reportViewer5.Visible = true; //muestra el reporte5
            reportViewer6.Visible = true; //muestra el reporte6
            this.reportViewer4.RefreshReport();//actualiza el reporte4
            this.reportViewer5.RefreshReport();//actualiza el reporte5
            this.reportViewer6.RefreshReport();//actualiza el reporte6
            break;
            case 45696: //velocidad = 45696
            reportViewer7.Visible = true; //muestra el reporte7
            reportViewer8.Visible = true; //muestra el reporte8
            reportViewer9.Visible = true; //muestra el reporte9
            this.reportViewer7.RefreshReport();//actualiza el reporte7
            this.reportViewer8.RefreshReport();//actualiza el reporte8
            this.reportViewer9.RefreshReport();//actualiza el reporte9
            break;
            case 143360: //velocidad = 143360
            reportViewer10.Visible = true; //muestra el reporte10
            reportViewer11.Visible = true; //muestra el reporte11
            reportViewer12.Visible = true; //muestra el reporte12
            this.reportViewer10.RefreshReport();//actualiza el reporte10
            this.reportViewer11.RefreshReport();//actualiza el reporte11
            this.reportViewer12.RefreshReport();//actualiza el reporte12
            break;
```

```
            }
        }

        private    void    comboBox1_SelectedValueChanged(object    sender,
EventArgs e)
        {
         NumMax     =     ((LimitarCanales)comboBox1.SelectedItem).Limite;
//obtiene el valor Limite de número de canales que puede tener una tecnología de tx
        }
    }
```

CLASE: LIMITARCANALES

```
class LimitarCanales
    {
        private int vlLimite=0; //variable para el limte del canal
        private string vlNombre="";//variable para el nombre del canal

        public LimitarCanales(string Nombre, int Capacidad)
        {
            vlLimite = Capacidad; //asigna un limite para los canales del AB
            vlNombre = Nombre; //asigna un valor a cada limite
        }

        public int Limite
        {
            get
            {
                return vlLimite; //retorna el valor del limite AB
            }
            set
            {
                vlLimite = value; //asigna el valor del limite
            }

        }
        public string Nombre
        {
            get
            {
                return vlNombre; //retorna vlNombre
            }
            set
            {
                vlNombre = value; //asigna el valor del Ancho de Banda
            }
        }
        public override string ToString()
        {
            return vlNombre; //retorna vlNombre
        }
    }
```

CLASE: CLASE RAMANODO

```
class RamaNodo
    {
        long vPeso; //variable para valor del peso del archivo
        int vNivel; //variable para el nivel del caracter leido
        int vChr; //variable para el valor del caracter leido

        string  vSimbolo; //variable para el valor del simbolo
        string vCodigo; //variable para el valor del codigo

        RamaNodo vRamaI; //variable que almacena la rama izquierda
        RamaNodo vRamaD; //variable que almacena la rama derecha

        Boolean vEs_Rama; //verifica si es rama o nodo

        public RamaNodo()
        {
        }
        public RamaNodo(RamaNodo RamaI, RamaNodo RamaD)
        {
            vPeso = RamaI.GetPeso() + RamaD.GetPeso();//suma la Frec de los
caracteres de RamasI y RamasD
            vSimbolo = RamaI.GetSimbolo() + RamaD.GetSimbolo();//junta los
ASCII de RamasI y RamasD
            vRamaI = RamaI; //valores de RamasI
            vRamaD = RamaD; //valores de RamasD
            vNivel = 0; //asigna 0 a vNivel
            vCodigo = "";//asigna vacio a vCodigo
            vEs_Rama = true; //asigna verdadero a vEs_Rama
            vChr = 0; //asigna 0 a vChar
            RamaI.BajarNivel("0");//llama a BajarNivel()
            RamaD.BajarNivel("1");//llama a BajarNivel()
        }

        public RamaNodo(long Peso,int Chr)
        {
            vPeso = Peso; //asigna el valor de Peso de la posicion del vector del
caracter leido
            vSimbolo = Convert.ToChar(Chr).ToString();//convierte en letra el
valor del vector
            vNivel = 0; //asigna 0 a vNivel
            vCodigo = "";//asigna vacio a vCodigo
            vEs_Rama = false; //asigna falso a vEs_Rama
            vChr = Chr; //asigna el valor del caracter a vChar
        }

        public long GetPeso()
        {
            return vPeso; //retorna el valor de vPeso
        }
        public int GetChr()
        {
            return vChr; // retorna el valor de vChar
        }
        public string GetCodigo()
        {
            return vCodigo; // retorna el valor de vCodigo
        }
        public string GetSimbolo()
        {
```

```
            return vSimbolo; // retorna el valor de vSimbolo
        }
        public int GetNivel()
        {
            return vNivel; // retorna el valor de vNivel
        }
        public bool GetEsRama()
        {
            return vEs_Rama; // retorna el valor de vEs_Rama
        }

        public void PlusNilvel()
        {
            vNivel++; //sube un nivel al arbol
        }
        public void SetCodigo(string Codigo)
        {
            vCodigo = Codigo; //asigna el valor del Codigo a vCodigo
        }
        private void BajarNivel(string Lado)
        {
            PlusNilvel(); //llama a PlusNivel()
            SetCodigo(Lado + vCodigo); //genera un código para cada rama
            if (vRamaI != null)
            {
                vRamaI.BajarNivel(Lado); //incrementa un nivel
            }
            if (vRamaD != null)
            {
                vRamaD.BajarNivel(Lado); //incrementa un nivel
            }
        }

    }
```

CLASE: CNTTU12

```
public partial class CtnTU12 : UserControl
    {
        public int cuentaTu=0; //contador de usuarios TU_12
        public TU_12[] ArrayTU12; //asigna ArrayTU12 como vector de tipo TU_12
        private CtnUserTU12 UsuariosTU12; //asignación de variable de tipo
CtnUsuerTU12

        public void fijarUsuarios(CtnUserTU12 Usuarios)
        {
            UsuariosTU12 = Usuarios; //asigna a UsuariosTU12 los componentes de la
interfaz de usuario TU12
        }

        private void AddStm1()
        {
            ArrayTU12[cuentaTu] = new TU_12();//inicializa ArrayTu12 como
instancia de TU_12
            ArrayTU12[cuentaTu].label67.Text = "STM1 - " +
cuentaTu.ToString();//asigna el número de la trama STM-1 al label
            flowLayoutPanel1.Controls.Add(ArrayTU12[cuentaTu]); //agrega al
layout los componentes
            cuentaTu++;//incrementa el contador de usuarios TU12
        }

        private void DelStm1()
        {
            cuentaTu--;//disminuye el contador de usuarios TU12
            ArrayTU12[cuentaTu].Dispose();//oculta la interfaz de usuario TU12
        }

        public void FijarCanal(int Stm1, int x, int y, int z, string
Usuario , Color MiColor)
        {
            ArrayTU12[Stm1].ArrayText[x, y, z].Text = Usuario; //Asigna el
nombre del canal al canal seleccionado por el usuario
            ArrayTU12[Stm1].ArrayText[x, y, z].BackColor = MiColor;
//asigna un color de usuario al canal TU12
            UsuariosTU12.ActualizarCanales();//llama a ActualizarCanales
        }
        public void LiberarCanal(int Stm1, int x, int y, int z)
        {
            ArrayTU12[Stm1].ArrayText[x, y, z].Text = "0000000";//asigna
00000000 al canal TU12
            ArrayTU12[Stm1].ArrayText[x, y, z].BackColor = Color.White;
//asigna blanco como color del canal TU12
            UsuariosTU12.ActualizarCanales();//llama a ActualizarCanales
        }

        public CtnTU12()
        {
            InitializeComponent();//inicializa los componentes de CTNTU12
        }

        private void numericUpDown1_ValueChanged(object sender, EventArgs
e)
        {
            if (numericUpDown1.Value > cuentaTu)
            {
                AddStm1();//llama a AddStm1
```

```
            }
            else
            {
                DelStm1();//llama a DelStm1
            }
            UsuariosTU12.ActualizarCanales();//actualiza los canales
        }

        private void CtnTU12_Load(object sender, EventArgs e)
        {
            ArrayTU12=  new  TU_12[200]; //inicializa  el  vector  ArrayTU12  como
instancia de TU_12
        }
    }
```

CLASE: CTNUSERTU12

```
public partial class CtnUserTU12 : UserControl
    {
        public CtnTU12 VGMedios; //variable global de CtnTU12, contiene variables de usuario
        public int UsuariosCtn = 0; ;//variable para control de usuarios
        public Usuario001[] ArraryUsuarios; //vector de tipo Usuario001 para almacenar datos de los usuarios

        private void AddUsuario()
        {
            ArraryUsuarios[UsuariosCtn] = new Usuario001();//inicializa la variable tipo vector ArraryUsuarios como instancia de Usuario001
            ArraryUsuarios[UsuariosCtn].label1.Text = "User " + UsuariosCtn.ToString();//asigna un número de usuario
            flowLayoutPanel1.Controls.Add(ArraryUsuarios[UsuariosCtn]); //agrega al layout los controles de usuario
            ArraryUsuarios[UsuariosCtn].FijarSubMedio(ref VGMedios); //llama a FijarSubMedio() en Usuario001 para definir si el medio es V1 o V2
            UsuariosCtn++;//incrementa el numero de usuarios
        }

        private void DelUsuario()
        {
            UsuariosCtn--;//disminuye el número de usuarios
            for (int i = 0; i < ArraryUsuarios[UsuariosCtn].listBox1.Items.Count; i++)++)
            {
                int S, x, y, z; //asignación de variables
                S = Convert.ToInt32(ArraryUsuarios[UsuariosCtn].listBox1.Items[i].ToString().Substring(5, 2)); //convierte a entero el indicador del número de trama
                x = Convert.ToInt32(ArraryUsuarios[UsuariosCtn].listBox1.Items[i].ToString().Substring(10, 1)); //convierte a entero el indicador de la posición x del vector de canales TU-12
                y = Convert.ToInt32(ArraryUsuarios[UsuariosCtn].listBox1.Items[i].ToString().Substring(11, 1)); //convierte a entero el indicador de la posición y del vector de canales TU-12
                z = Convert.ToInt32(ArraryUsuarios[UsuariosCtn].listBox1.Items[i].ToString().Substring(12, 1)); //convierte a entero el indicador de la posición z del vector de canales TU-12
ArraryUsuarios[UsuariosCtn].MyMedioTU12.LiberarCanal(S, x, y, z); // libera el canal TU_12
            }
            ArraryUsuarios[UsuariosCtn].Dispose();//oculta la interfaz de usuario
        }

        public void FijarMedio(ref CtnTU12 Medio)
        {
            VGMedios = Medio; //asigna a VGMedios los componentes de CtnTU_12
        }

        public void ActualizarCanales()
        {
            for (int i = 0; i < UsuariosCtn; i++)
            {
```

```
                ArrarayUsuarios[i].listBox1.Items.Clear();//limpia los items
del list box
                ArrarayUsuarios[i].listBox2.Items.Clear();//limpia los items
del list box
                for (int j = 0; j < VGMedios.cuentaTu; j++)
                {
                    string AuD = "";//asignación de variable
                    if (j < 10) { AuD = "0"; }
                    for (int x = 0; x < 3; x++)//contador para K
                    {
                        for (int y = 0; y < 7; y++)//contador para L
                        {
                            for (int z = 0; z < 3; z++)//contador para M
                            {
                                if (VGMedios.ArrayTU12[j].ArrayText[x, y,
z].Text == "0000000")//asigna 00000000 para las posiciones x,y,z
                                {
ArrarayUsuarios[i].listBox2.Items.Add("STM1-" + AuD + j.ToString() + " TU"
+ x.ToString() + y.ToString() + z.ToString());//agrega en el list box las
posiciones de los TU_12
                                }
                                else
                                {
                                    if
(VGMedios.ArrayTU12[j].ArrayText[x,          y,          z].Text          ==
ArrarayUsuarios[i].label1.Text) //asigna el label canal del TU_12 seleccionado por el
usuario
                                    {
ArrarayUsuarios[i].listBox1.Items.Add("STM1-" + AuD + j.ToString() + " TU"
+ x.ToString() + y.ToString() + z.ToString());//agrega al listbox 1 las
posiciones de los TU_12
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }

        public CtnUserTU12()
        {
            InitializeComponent();//inicializa los componentes
            ArrarayUsuarios = new Usuario001[200]; //inicializa ArrarayUsuarios
como instancia de Usuario001
        }

        private void numericUpDown1_ValueChanged(object sender, EventArgs
e)
        {
            if (numericUpDown1.Value > UsuariosCtn)
            {
                AddUsuario();//llama a AddUsuario
            }
            else
            {
                DelUsuario();//llama a DelUsuario
            }
            ActualizarCanales();//llama a ActualizarCanales
        }
    }
```

CLASE: E1

```
public partial class E1 : UserControl
    {
        Boolean V2; //define a V2 como booleano
        public Label[] Canales; //define a Canales como vector de tipo Label

        public E1()
        {
            InitializeComponent();//inicializa los componentes
            Asociar();//llama a asociar
        }

        private void Asociar()
        {
            Canales = new Label[32]; //define Canales como un vectorde 32
posiciones de tipo Label
            Canales[0] = label0; //asigna el label0 a un canal de la trama
            Canales[1] = label1; //asigna el label1 a un canal de la trama
            Canales[2] = label2; //asigna el label2 a un canal de la trama
            Canales[3] = label3; //asigna el label3 a un canal de la trama
            Canales[4] = label4; //asigna el label4 a un canal de la trama
            Canales[5] = label5; //asigna el label5 a un canal de la trama
            Canales[6] = label6; //asigna el label6 a un canal de la trama
            Canales[7] = label7; //asigna el label7 a un canal de la trama
            Canales[8] = label8; //asigna el label8 a un canal de la trama
            Canales[9] = label9; //asigna el label9 a un canal de la trama
            Canales[10] = label10; //asigna el label10 a un canal de la trama
            Canales[11] = label11; //asigna el label11 a un canal de la trama
            Canales[12] = label12; //asigna el label12 a un canal de la trama
            Canales[13] = label13; //asigna el label13 a un canal de la trama
            Canales[14] = label14; //asigna el label14 a un canal de la trama
            Canales[15] = label15; //asigna el label15 a un canal de la trama
            Canales[16] = label16; //asigna el label16 a un canal de la trama
            Canales[17] = label17; //asigna el label17 a un canal de la trama
            Canales[18] = label18; //asigna el label18 a un canal de la trama
            Canales[19] = label19; //asigna el label19 a un canal de la trama
            Canales[20] = label20; //asigna el label20 a un canal de la trama
            Canales[21] = label21; //asigna el label21 a un canal de la trama
            Canales[22] = label22; //asigna el label22 a un canal de la trama
            Canales[23] = label23; //asigna el label23 a un canal de la trama
            Canales[24] = label24; //asigna el label24 a un canal de la trama
            Canales[25] = label25; //asigna el label25 a un canal de la trama
            Canales[26] = label26; //asigna el label26 a un canal de la trama
            Canales[27] = label27; //asigna el label27 a un canal de la trama
            Canales[28] = label28; //asigna el label28 a un canal de la trama
            Canales[29] = label29; //asigna el label29 a un canal de la trama
            Canales[30] = label30; //asigna el label30 a un canal de la trama
            Canales[31] = label31; //asigna el label31 a un canal de la trama
        }

        public E1(String NumeroE1)
        {
            InitializeComponent();//inicializa los componentes al recibir el
número de trama E1
            label32.Text = NumeroE1; //asigna al label el valor del número de
trama
            Asociar();//llama a asociar en E1
        }
```

```
        public void SetV2(Boolean Valor)
        {
            V2 = Valor; //asigna el valor de versión de E1 v1 o v2
        }

        public void SetValue(int Pos, string Value)
        {
            switch (Pos) //recibe la posición del canal de la trama E1
            {
                case 0:
                    label0.Text = Value;//asigna a label0 el valor del byte leido
                    break;
                case 1:
                    label1.Text = Value;//asigna a label1 el valor del byte leido
                    break;
                case 2:
                    label2.Text = Value;//asigna a label2 el valor del byte leido
                    break;
                case 3:
                    label3.Text = Value;//asigna a label4 el valor del byte leido
                    break;
                case 4:
                    label4.Text = Value;//asigna a label5 el valor del byte leido
                    break;
                case 5:
                    label5.Text = Value;//asigna a label5 el valor del byte leido
                    break;
                case 6:
                    label6.Text = Value //asigna a label6 el valor del byte leido
                    break;
                case 7:
                    label7.Text = Value;//asigna a label7 el valor del byte leido
                    break;
                case 8:
                    label8.Text = Value;//asigna a label8 el valor del byte leido
                    break;
                case 9:
                    label9.Text = Value;//asigna a label9 el valor del byte leido
                    break;
                case 10:
                    label10.Text = Value;/asigna a label10 el valor del byte
leido
                    break;
                case 11:
                    label11.Text = Value;/asigna a label11 el valor del byte
leido
                    break;
                case 12:
                    label12.Text = Value;/asigna label12 el valor del byte leido
                    break;
                case 13:
                    label13.Text = Value;/asigna label13 el valor del byte leido
                    break;
                case 14:
                    label14.Text = Value;/asigna label14 el valor del byte leido
                    break;
                case 15:
                    label15.Text = Value;/asigna label15 el valor del byte leido
                    break;
                case 16:
```

```
                label16.Text = Value;/asigna label16 el valor del byte leido
                break;
            case 17:
                label17.Text = Value;/asigna label17 el valor del byte leido
                break;
            case 18:
                label18.Text = Value;//asigna label18 el valor del byte leido
                break;
            case 19:
                label19.Text = Value;//asigna label18 el valor del byte leido
                break;
            case 20:
                label20.Text = Value;//asigna label20 el valor del byte leido
                break;
            case 21:
                label21.Text = Value;//asigna label21 el valor del byte leido
                break;
            case 22:
                label22.Text = Value;//asigna label22 el valor del byte leido
                break;
            case 23:
                label23.Text = Value;//asigna label23 el valor del byte leido
                break;
            case 24:
                label24.Text = Value;//asigna label24 el valor del byte leido
                break;
            case 25:
                label25.Text = Value;//asigna label25 el valor del byte leido
                break;
            case 26:
                label26.Text = Value;//asigna label26 el valor del byte leido
                break;
            case 27:
                label27.Text = Value;//asigna label27 el valor del byte leido
                break;
            case 28:
                label28.Text = Value;//asigna label28 el valor del byte leido
                break;
            case 29:
                label29.Text = Value;//asigna label29 el valor del byte leido
                break;
            case 30:
                label30.Text = Value;//asigna label30 el valor del byte leido
                break;
            case 31:
                label31.Text = Value;//asigna label32 el valor del byte leido
                break;
        }
    }
    public void SetColor(int Pos, Color Value)
    {
        switch (Pos)
        {
            case 0:
                label0.BackColor = Value;//asigna a label0 el color elegido
                break;
            case 1:
                label1.BackColor = Value;//asigna a label1 el color elegido
                break;
            case 2:
```

```
        label2.BackColor = Value;//asigna a label2 el color elegido
        break;
    case 3:
        label3.BackColor = Value;//asigna a label3 el color elegido
        break;
    case 4:
        label4.BackColor = Value;//asigna a label4 el color elegido
        break;
    case 5:
        label5.BackColor = Value;//asigna a label5 el color elegido
        break;
    case 6:
        label6.BackColor = Value;//asigna a label6 el color elegido
        break;
    case 7:
        label7.BackColor = Value;//asigna a label7 el color elegido
        break;
    case 8:
        label8.BackColor = Value;//asigna a label8 el color elegido
        break;
    case 9:
        label9.BackColor = Value;//asigna a label9 el color elegido
        break;
    case 10:
        label10.BackColor = Value;/asigna a label10 el color elegido
        break;
    case 11:
        label11.BackColor = Value;/asigna a label11 el color elegido
        break;
    case 12:
        label12.BackColor = Value;/asigna a label12 el color elegido
        break;
    case 13:
        label13.BackColor = Value;/asigna a label13 el color elegido
        break;
    case 14:
        label14.BackColor = Value;/asigna a label14 el color elegido
        break;
    case 15:
        label15.BackColor = Value;/asigna a label15 el color elegido
        break;
    case 16:
        label16.BackColor = Value;/asigna a label16 el color elegido
        break;
    case 17:
        label17.BackColor = Value;/asigna a label17 el color elegido
        break;
    case 18:
        label18.BackColor = Value;/asigna a label18 el color elegido
        break
    case 19:
        label19.BackColor = Value;/asigna a label19 el color elegido
        break;
    case 20:
        label20.BackColor = Value;/asigna a label20 el color elegido
        break;
    case 21:
        label21.BackColor = Value;/asigna a label21 el color elegido
        break;
    case 22:
```

```
                    label22.BackColor = Value;/asigna a label22 el color elegido
                    break;
                case 23:
                    label23.BackColor = Value;/asigna a label23 el color elegido
                    break;
                case 24:
                    label24.BackColor = Value;/asigna a label24 el color elegido
                    break;
                case 25:
                    label25.BackColor = Value;/asigna a label25 el color elegido
                    break;
                case 26:
                    label26.BackColor = Value;/asigna a label26 el color elegido
                    break;
                case 27:
                    label27.BackColor = Value;/asigna a label27 el color elegido
                    break;
                case 28:
                    label28.BackColor = Value;/asigna a label28 el color elegido
                    break;
                case 29:
                    label29.BackColor = Value;/asigna a label29 el color elegido
                    break;
                case 30:
                    label30.BackColor = Value;/asigna a label30 el color elegido
                    break;
                case 31:
                    label31.BackColor = Value;/asigna a label31 el color elegido
                    break;
            }
        }

    }
```

CLASE: FORM1

```
public partial class Form1 : Form
    {
        E1[] Matriz;//declara a Matriz como variable de tipo E1
        int ArchivoPeso = 0; //declara a la variable ArchivoPesos como entero
        int x, y, l; //declara a x,y,1 como enteros

        public Form1()
        {
            InitializeComponent();//inicializa los componentes del Form1
        }

        public void AgregarValor(String Valor)
        {
            int res = 0; //inicializa la variable res
            if (l == 0 || x==32)
            {
                Matriz[l] = new E1("E1("+(l+1).ToString()+")");//Matriz[l]
instancia de E1 almacena los canales de la trama E1
            if (l < 50)
                {
                    flowLayoutPanel1.Controls.Add(Matriz[l]); // carga los
controles en el flowlayoutpanel
                }
                else
                {
                    res = 1; //asigna el valor de 1 a la variable res
                }
                l++; x = 1; y = l - 1; //asigna los valore a las variables
                Matriz[y].SetValue(0, Cabeza());//asigna a la posición 0 de
la trama el valor de cabecera
                Matriz[y].SetValue(16, Verificar());//asigna a la posición
16 de la trama el valor de control
            }
            if (x == 16)
            {
                x++;//suma a x uno
            }
            Matriz[y].SetValue(x, Valor); //ubica los valores en los canales
de la trama
            x++;//aumenta el contador de caracteres
            return res; //retorna el valor de res
        }

        public string Cabeza()
        {
            return  IntToBinary(201);  //retorna  el  valor  de  la  división
binaria que es el valor de cabecera
        }

        public string Verificar()
        {
            return "11110000";//valor de control
        }

        public string IntToBinary(int Val)
        {
            if (Val > 0) //si el valor pasado por IntToBinary8 es mayor a 0
            {
```

```
                int a; //declara la variable a
                a=Val/2;  //en  a  se  almacena  el  resultado  de  la  división
binaria para transformar el valor del caracter leido de decimal a binario
                return (IntToBinary(a) + (Val % 2).ToString());//retorna
como string el valor de la división para compararlo nuevamente hasta que sea 0
            }
            else
            {
                return "";//caso contrario retorna vacio
            }
        }

        public string IntToBinary8(int Val)
        {
            string low = IntToBinary(Val); //almacena la división binaria
            int  lon  =  low.Length; //almacena en lon el tamaño de bits de la
división
            for (int i=0;i<(8-lon);i++)
            {
                low = "0" + low; //aumenta un 0 a low para completsr el byte
            }
            return low; //retorna low
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            Matriz  =  new  E1[10240];//asigna  a  Matriz  un  vector  de  10240
posicines
            x = 0; y = 0; l = 0;//inicializa en 0 los valores citados
        }

        private void button1_Click(object sender, EventArgs e)
        {
            ArchivoPeso = 0; //inicializa la variable ArchivoPeso en 0
            FO1.ShowDialog();//muestra una ventana de selección de archivos
            if (System.IO.File.Exists(FO1.FileName))//verifica si existe el
archivo seleccionado
            {
                using         (System.IO.StreamReader         sr         =
System.IO.File.OpenText(FO1.FileName)) //abre el archivo de texto y define
una variable de lectura
                {
                    int s=0; //inicializa la variable en 0
                    while((s = sr.Read()) != -1)
                    {
                    if (AgregarValor(IntToBinary8(s)) == 0)
                        {
                           ArchivoPeso++;//Aumenta el tamaño del archivo
                        }
                        else
                        {
                            MessageBox.Show("Archivo demasiado grande
para mostrar. Por motivos didácticos solo se mostrará una parte");
//mensaje al usuario
                            break; //forza para salir
                        }
                     }
                  }
                label5.Text  =  ArchivoPeso.ToString()  +  "  bytes";//coloca
en el label el tamaño del archivo
            }
```

```
        }

private void button2_Click(object sender, EventArgs e)
        {
            ArchivoPeso = 0; //inicializa la variable en 0
            l = 0; //inicializa la variable en 0
            y = 0; //inicializa la variable en 0
            Matriz = new E1 [200]; //define una matriz de tipo E1
            for (l = 0; l < 50; l++)
            {
                flowLayoutPanel1.Controls.Remove(Matriz[l]); //remueve los
controles del FlowLayout
            }
            l = 0; //inicializa la variable en 0
            flowLayoutPanel1.Controls.Clear();//limpia los controles del
FlowLayout
            label5.Text = ArchivoPeso.ToString() + " bytes";//asigna al
label el valor del peso del archivo
            MessageBox.Show("Se limpiaran los datos. Puede seleccionar
otro archivo");//mensaje al usuario
        }

}
```

CLASE: FORM2

```
public partial class Form2 : Form
    {
        UserE1[] ListaUsuarios; //define a ListaUsuarios como vector de tipo
UserE1
        private string FileFlash="";//variable FileFlash como string
        int TotalUsuarios; //variable TotalUsuarios como entero

        public void ActualizarArreglo()
        {
            for (int i = 0; i < TotalUsuarios; i++)
            {
                ListaUsuarios[i].ActualizarCanales();//actualiza       los
canales de los usuarios eliminados
            }
        }

        public Form2(ref AxAgentObjects.AxAgent Agente)
        {
            InitializeComponent();//inicializa los componentes
        }

        private void Form2_Load(object sender, EventArgs e)
        {

            comboBox2.SelectedIndex = 0; //aisgna un 0 a la variable
            e11.SetV2(true); //define como versión 2 V2
            e11.SetValue( 0, "11011011");//define el valor de cabecera para
la posición 0 de la trama
            e11.SetValue(16, "11011011");//define el valor de control para
la posición 16 de la trama
            ListaUsuarios  =  new  UserE1[32];  //define  un  vector  de  32
posiciones tipo UserE1
            TotalUsuarios = 0; //inicializa en 0 la variable
            string                          appPath                        =
Path.GetDirectoryName(Application.ExecutablePath);  //define  appPath  como
ruta de ejecución de la aplicación
            FileFlash = appPath + "\\Simul_E1.swf";//define FileFlash como
ruta de ejecución de las animaciones de tipo Flash
            if (axShockwaveFlash1.Movie != FileFlash)
            {
                axShockwaveFlash1.Movie = FileFlash; //prepara la ruta de
la animación Flash
                axShockwaveFlash1.Play();//ejecuta la animación de Flash
            }

        }

        private void e11_Resize(object sender, EventArgs e)
        {
            ActualizarArreglo();//llama a ActualizarCanales
        }

        private void numericUpDown2_ValueChanged(object sender, EventArgs
e)
        {

            if (TotalUsuarios < numericUpDown2.Value)
            {
```

```
                ListaUsuarios[TotalUsuarios]     =     new     UserE1(e11,
(TotalUsuarios  +  1).ToString());//inicializa  el  vector  ListaUsuarios  como
instancia de UserE1 agrega una trama E1 a la interfaz de usuario
flowLayoutPanel1.Controls.Add(ListaUsuarios[TotalUsuarios]);  //agrega  al
layout los componentes de usuario
                TotalUsuarios++;//incrementa el contador de usuarios
            }
            else
            {
                TotalUsuarios--;//disminuye el contador de usuarios
                ListaUsuarios[TotalUsuarios].LiberarCanales();//libera  un
canal de un usuario eliminado
                ListaUsuarios[TotalUsuarios].Dispose();//oulta  la  interfaz
del usuario eliminado
            }
        }

        private void numericUpDown1_ValueChanged(object sender, EventArgs
e)
        {
            label5.Text    =    numericUpDown1.Value.ToString()+    "    Km
";//muestra la distancia de acuerdo al numericUpDown
        }


    }
```

CLASE: PROGRAM

```
static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new  MDIParent1());//inicializa  la  aplicación
desde MDIParent1
        }
    }
```

CLASE: MDIPARENT

```
public partial class MDIParent1 : Form
    {
        private int childFormNumber = 0; //declara la variable
childFormNumber

        public MDIParent1()
        {
            InitializeComponent();//inicializa los componentes
        }

        private void ShowNewForm(object sender, EventArgs e)
        {
            Form childForm = new Form();//variable chilForm como instancia
de Form
            childForm.MdiParent = this; //asigna a childForm la ventana
MDIParent
            childForm.Text = "Window " + childFormNumber++;//cuenta los
formularios
            childForm.Show();//muestra el formulario
        }

        private void ExitToolsStripMenuItem_Click(object sender,
EventArgs e)
        {
            this.Close();//cierra el formulario abierto
        }

        private void arcivoEnE1ToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            Form1 f1; // declara a f1 como variable de tipo Form1
            f1 = new Form1();//instancia de form1
            f1.ShowDialog();//muestra Form1
        }

        private void variosUsuariosToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            Form2 f2; //declara a f2 como variable de tipo Form2
            f2 = new Form2(ref axAgent1); //instancia de form2
            f2.ShowDialog();//muestra el formulario 2
        }

        private void tU12ToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            HufmanV2.STM1.STM1_TU12 V2 = new
HufmanV2.STM1.STM1_TU12();//variable V2 instancia de tipo STM1_TU12
            V2.ShowDialog();//muestra el formulario para TU_12
        }

        private void vCToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            FrmVariosE1V2 V2 = new FrmVariosE1V2("AU3");//variable V2
instancia de FrmVariosE1V2
            V2.ShowDialog();//muestra el formulario para AU3
        }
```

```
        private void vC4ToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            FrmVariosE1V2 V2 = new FrmVariosE1V2("AU4");//variable V2
instancia de FrmVariosE1V2
            V2.ShowDialog();//muestra el formulario para AU4
        }

        private void pruebaV2ToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            FrmVariosE1V2 V2 = new FrmVariosE1V2("E1");//variable V2
instancia de FrmVariosE1V2
            V2.ShowDialog();//muestra el formulario para E1
        }
    }
```

CLASE: TU_12

```
public partial class TU_12 : UserControl
    {
        public TextBox[,,] ArrayText; //variable tipo vector de 3 posiciones

        private void AsociarCanales()
        {
            ArrayText = new TextBox[3, 7, 3]; //inicializa el vector de 3
posiciones

            ArrayText[0, 0, 0] = textBox1; //inicializa el vector en 0,0,0
            ArrayText[0, 0, 1] = textBox2; //inicializa el vector en 0,0,1
            ArrayText[0, 0, 2] = textBox3; //inicializa el vector en 0,0,2
            ArrayText[0, 1, 0] = textBox4; //inicializa el vector en 0,1,0
            ArrayText[0, 1, 1] = textBox5; //inicializa el vector en 0,1,1
            ArrayText[0, 1, 2] = textBox6; //inicializa el vector en 0,1,2
            ArrayText[0, 2, 0] = textBox7; //inicializa el vector en 0,2,0
            ArrayText[0, 2, 1] = textBox8; //inicializa el vector en 0,2,1
            ArrayText[0, 2, 2] = textBox9; //inicializa el vector en 0,2,2
            ArrayText[0, 3, 0] = textBox10; //inicializa el vector en 0,3,0
            ArrayText[0, 3, 1] = textBox11; //inicializa el vector en 0,3,1
            ArrayText[0, 3, 2] = textBox12; //inicializa el vector en 0,3,2
            ArrayText[0, 4, 0] = textBox13; //inicializa el vector en 0,4,0
            ArrayText[0, 4, 1] = textBox14; //inicializa el vector en 0,4,1
            ArrayText[0, 4, 2] = textBox15; //inicializa el vector en 0,4,2
            ArrayText[0, 5, 0] = textBox16; //inicializa el vector en 0,5,0
            ArrayText[0, 5, 1] = textBox17; //inicializa el vector en 0,5,1
            ArrayText[0, 5, 2] = textBox18; //inicializa el vector en 0,5,2
            ArrayText[0, 6, 0] = textBox19; //inicializa el vector en 0,6,0
            ArrayText[0, 6, 1] = textBox20; //inicializa el vector en 0,6,1
            ArrayText[0, 6, 2] = textBox21; //inicializa el vector en 0,6,2

            ArrayText[1, 0, 0] = textBox22; //inicializa el vector en 1,0,0
            ArrayText[1, 0, 1] = textBox29; //inicializa el vector en 1,0,1
            ArrayText[1, 0, 2] = textBox36; //inicializa el vector en 1,0,2
            ArrayText[1, 1, 0] = textBox23; //inicializa el vector en 1,1,0
            ArrayText[1, 1, 1] = textBox30; //inicializa el vector en 1,1,1
            ArrayText[1, 1, 2] = textBox37; //inicializa el vector en 1,1,2
            ArrayText[1, 2, 0] = textBox24; //inicializa el vector en 1,2,0
            ArrayText[1, 2, 1] = textBox31; //inicializa el vector en 1,2,1
            ArrayText[1, 2, 2] = textBox38; //inicializa el vector en 1,2,2
            ArrayText[1, 3, 0] = textBox25; //inicializa el vector en 1,3,0
            ArrayText[1, 3, 1] = textBox32; //inicializa el vector en 1,3,1
            ArrayText[1, 3, 2] = textBox39; //inicializa el vector en 1,3,2
            ArrayText[1, 4, 0] = textBox26; //inicializa el vector en 1,4,0
            ArrayText[1, 4, 1] = textBox33; //inicializa el vector en 1,4,1
            ArrayText[1, 4, 2] = textBox40; //inicializa el vector en 1,4,2
            ArrayText[1, 5, 0] = textBox27; //inicializa el vector en 1,5,0
            ArrayText[1, 5, 1] = textBox34; //inicializa el vector en 1,5,1
            ArrayText[1, 5, 2] = textBox41; //inicializa el vector en 1,5,2
            ArrayText[1, 6, 0] = textBox28; //inicializa el vector en 1,6,0
            ArrayText[1, 6, 1] = textBox35; //inicializa el vector en 1,6,1
            ArrayText[1, 6, 2] = textBox42; //inicializa el vector en 1,6,2

            ArrayText[2, 0, 0] = textBox43; //inicializa el vector en 2,0,0
            ArrayText[2, 0, 1] = textBox50; //inicializa el vector en 2,0,1
            ArrayText[2, 0, 2] = textBox57; //inicializa el vector en 2,0,2
            ArrayText[2, 1, 0] = textBox44; //inicializa el vector en 2,1,0
            ArrayText[2, 1, 1] = textBox51; //inicializa el vector en 2,1,1
```

```
            ArrayText[2, 1, 2] = textBox58; //inicializa el vector en 2,1,2
            ArrayText[2, 2, 0] = textBox45; //inicializa el vector en 2,2,0
            ArrayText[2, 2, 1] = textBox52; //inicializa el vector en 2,2,1
            ArrayText[2, 2, 2] = textBox59; //inicializa el vector en 2,2,2
            ArrayText[2, 3, 0] = textBox46; //inicializa el vector en 2,3,0
            ArrayText[2, 3, 1] = textBox53; //inicializa el vector en 2,3,1
            ArrayText[2, 3, 2] = textBox60; //inicializa el vector en 2,3,2
            ArrayText[2, 4, 0] = textBox47; //inicializa el vector en 2,4,0
            ArrayText[2, 4, 1] = textBox54; //inicializa el vector en 2,4,1
            ArrayText[2, 4, 2] = textBox61; //inicializa el vector en 2,4,2
            ArrayText[2, 5, 0] = textBox48; //inicializa el vector en 2,5,0
            ArrayText[2, 5, 1] = textBox55; //inicializa el vector en 2,5,1
            ArrayText[2, 5, 2] = textBox62; //inicializa el vector en 2,5,2
            ArrayText[2, 6, 0] = textBox49; //inicializa el vector en 2,6,0
            ArrayText[2, 6, 1] = textBox56; //inicializa el vector en 2,6,1
            ArrayText[2, 6, 2] = textBox63; //inicializa el vector en 2,6,2
        }

        public TU_12()
        {
            InitializeComponent();//inicializa los componentes de TU_12
        }

        private void TU_12_Load(object sender, EventArgs e)
        {
            AsociarCanales();//llama a AsociarCanales
        }
    }
```

CLASE: USER E1

```
public partial class UserE1 : UserControl
    {
        E1 MyE1; //variable MyE1 de tipo E1
        Huffman ArchTemp; //variable ArchTemp de tipo Huffman
        int Factor = 1; //variable Factor de tipo entero

        private bool ArchivoExiste(string Archivo)
        {
            return true; //retorna si el archivo existe
        }

        public UserE1()
        {
            InitializeComponent();//inicializa los componentes de UserE1
        }

        public UserE1(E1 Medio, string Nombre)
        {
            InitializeComponent();//inicializa los componentes
            label1.Text = "Usuario " + Nombre; //asigna a Label1 el numero de
tramas E1
            MyE1 = Medio; //variable que contiene los componentes del form2
            ActualizarCanales();//llama a ActualizarCanales
            ArchTemp = new Huffman();//declara a ArchTemp como instancia de Huffman
        }

        public void LlenarCanales()
        {
            for (int i = 0; i < 6; i++)
            {

                comboBox1.Items.Add((Math.Pow(2, i) * 64).ToString() + "
MB");//asigna al combobox1 los canales del E1
            }
        }

        public void ActualizarCanales()
        {
            if (MyE1!=null)
            {
                listBox3.Items.Clear();//borra la lista3
                listBox2.Items.Clear();//borra la lista2
                for (int i=0;i<32;i++)                {
                    if (MyE1.Canales[i].Text=="00000000")
                    {
                        listBox3.Items.Add("Canal           "           +
i.ToString());//agrega el canal disponible en el listado de canales
                    }
                    if (MyE1.Canales[i].Text == label1.Text)
                    {
                        listBox2.Items.Add("Canal           "           +
i.ToString());//agrega el canal seleccionado al listado de canales ocupados
                    }
                }
            }
        }

        private void panel1_DoubleClick(object sender, EventArgs e)
```

```
        {
            CD1.ShowDialog();//muestra un cuadro de color
            panel1.BackColor = CD1.Color; //asigna al panel el color seleccionado
        }

        private void button1_Click(object sender, EventArgs e)
        {
            OFD1.ShowDialog();//abre una ventana de selección de archivos
            for (int i = 0; i < OFD1.FileNames.Length; i++)
            {
                if (ArchivoExiste(OFD1.FileNames[i]))
                {
                    listBox1.Items.Add(OFD1.FileNames[i]);  //agrega  a  la
lista el nombre del archivo seleccionado
                }
            }
        }

        private void UserE1_Load(object sender, EventArgs e)
        {
            LlenarCanales();//llama a LlenarCanales
        }

        private     void     listBox3_MouseDoubleClick(object     sender,
MouseEventArgs e)
        {
            if        (listBox2.Items.Count        <        Math.Pow(2,
comboBox1.SelectedIndex))
            {
                int vl_canal; //variable vl_canal como entero
                vl_canal                                                =
Convert.ToInt16(listBox3.SelectedItem.ToString().Substring(listBox3.Selec
tedItem.ToString().Length - 2, 2)); //obtiene el número del canal de la trama
                MyE1.Canales[vl_canal].Text   =   label1.Text;   //asigna   el
número del canal al label del E1
                MyE1.Canales[vl_canal].BackColor    =    panel1.BackColor;
//asigna el color del panel al canal seleccionado en la trama E1
                MyE1.Width = MyE1.Width + Factor; //obtiene la longitud de MyE1
                Factor = Factor * (-1); //aisgna 1 a Factor
            }
        }

        private     void     listBox2_MouseDoubleClick(object     sender,
MouseEventArgs e)
        {
            if (listBox2.SelectedItems.Count > 0)
            {
                int vl_canal; //variable vl_canal como entero
                vl_canal                                                =
Convert.ToInt16(listBox2.SelectedItem.ToString().Substring(listBox2.Selec
tedItem.ToString().Length - 2, 2)); //obtiene el número del canal de la trama
                MyE1.Canales[vl_canal].Text = "00000000";//asigna 00000000 al
label del E1
                MyE1.Canales[vl_canal].BackColor  =  Color.White;  //asigna
blanco como color del canal del E1
                MyE1.Width = MyE1.Width + Factor; //obtiene la longitud de MyE1
                Factor = Factor * (-1); //asigna 1 a Factor
            }
        }

        public void LiberarCanales()
```

```
        {
            for (int i = 0; i < listBox2.Items.Count; i++)
            {
                int vl_canal; //variable vl_canal de tipo entero
                vl_canal                                                =
Convert.ToInt16(listBox2.Items[i].ToString().Substring(listBox2.Items[i].
ToString().Length - 2, 2)); //obtiene los sub índices de la trama
                MyE1.Canales[vl_canal].Text = "00000000";//asigna 00000000 al
canal de la trama E1
                MyE1.Canales[vl_canal].BackColor  =  Color.White;  //asigna
blanco como color del canal
            }
            MyE1.Width = MyE1.Width + Factor; //obtiene la longitud de MyE1
            Factor = Factor * (-1); //asigna 1 a Factor
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (listBox1.SelectedItems.Count > 0)
            {
                listBox1.Items.Remove(listBox1.SelectedItem); //remueve del
listbox
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {

            string                          appPath                         =
Path.GetDirectoryName(Application.ExecutablePath); //captura el Path del Arc
            TBL_EstadisticasTableAdapter.Connection.ConnectionString    =
"Provider=Microsoft.Jet.OLEDB.4.0;Data    Source="    +    appPath    +
"\\DBEstadisticas.mdb;Persist Security Info=True";//detalle de la conexión
            if (listBox1.Items.Count > 0)
            {
                // borrar datos

                string  strConn  =  "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source="+appPath+"\\DBEstadisticas.mdb";//variable de conexión
                OleDbConnection  myAccConection  =  null;  //variable  de  tipo
OleDbConnection
                string strCommand; //variable para la acción SQL
                strCommand   =   "DELETE   FROM   TBL_Estadisticas   WHERE
Est_Usuario='" + label1.Text + "'";//comando para borrar los campos de la BD
                try
                {
                    myAccConection    =    new    OleDbConnection(strConn);
//define una acción de conexión
                    myAccConection.Open();//abre la conexión a la BD

                    OleDbCommand myCommand = new OleDbCommand(strCommand,
myAccConection); //instancia de OleDbCommand para la variable myCommand
                    myCommand.ExecuteNonQuery();//ejecución  de  la  instrucción
SQL
                    myAccConection.Close();//cierra la conexión a la BD
                }
                catch
                {
                }
                for (int i = 0; i < listBox1.Items.Count; i++)
                {
```

```
                    // definir variables

ArchTemp.CalcularArchivoPesos(listBox1.Items[i].ToString());/llama a
CalcularArchivoPesos
                    ArchTemp.CrearArbol();//llama a CrearArbol
                    string vl_Nombre, vl_Usuario, vl_Medio; //asignación de
variables
                    long vl_Peso; //asignación de vaariables
                    Int32 vl_zip, vl_tabla, vl_distancia, vl_AnchoBanda;
//asignación de variables
                    double vl_velocidad; //asignación de variables
                    // Calcular valores
                    vl_Nombre = listBox1.Items[i].ToString();//almacena el
nombre del archivo seleccionado
                    vl_Peso = ArchTemp.TamañoArchivo();//almacena el peso del
archivo
                    vl_zip=ArchTemp.PesoZip();//almacena el peso de archivo
comprimido
                    vl_tabla = ArchTemp.PesoTabla();//asigna el peso de la
tabla
                    vl_Usuario = label1.Text; //asigna el usuario al textbox
                    vl_Medio     =     this.ParentForm.Controls[5].Text;
//selecciona el medio
                    vl_velocidad = 0; //asigna la velocidad a 0
                    switch (this.ParentForm.Controls[5].Text)
                    {
                        case "Radio Frecuencia":
                            vl_velocidad = 0.00000003; //valor de velocidad
                            break;
                        case "Cobre":
                            vl_velocidad = 0.000000023;//valor de velocidad
                            break;
                        case "Fibra Optica":
                            vl_velocidad = 0.00000002; //valor de velocidad
                            break;
                    }
                    vl_distancia                                          =
Convert.ToInt32(this.ParentForm.Controls[6].Text); //asigna el vaor de la
distancia
                    vl_AnchoBanda = listBox2.Items.Count * 64; //asigna el
ancho de banda

                    // Isertar valores en tabla
                    strCommand=    "INSERT    INTO    TBL_Estadisticas    (
Est_nombre_arhivo,  Est_peso,  Est_Comprimido,  Est_Tabla,  Est_Usuario,
Est_Medio, Est_VelocidadT, Est_Distancia, Est_AnchoBanda ) values ('" +
vl_Nombre + "'," + vl_Peso + "," + vl_zip + "," + vl_tabla + ",'" +
vl_Usuario + "','" + vl_Medio + "'," + vl_velocidad + "," + vl_distancia
+ "," + vl_AnchoBanda + ")";//comando de inserción de datos en la base
                    try
                    {
                        myAccConection.Open();//abre la base de datos
                        OleDbCommand        myCommand        =        new
OleDbCommand(strCommand, myAccConection); //establece un comando de ejecución
                        myCommand.ExecuteNonQuery();//ejecuta el comando
                        myAccConection.Close();//cierra la conexión a la base
de datos
                    }
                    catch
                    {
                        label1.Text = strCommand; //manejo de errores
                    }
```

```
            }
        }

this.TBL_EstadisticasTableAdapter.FillByUsuario(this.DBEstadisticasDataSe
t.TBL_Estadisticas, label1.Text); //revisa los datos de la base de datos
            this.reportViewer1.RefreshReport();//muestra   los   datos   en   el
reporte1
            this.reportViewer2.RefreshReport();//muestra   los   datos   en   el
reporte2
            this.reportViewer3.RefreshReport();//muestra   los   datos   en   el
reporte3
        }

        private   void   comboBox1_SelectedIndexChanged(object   sender,
EventArgs e)
        {

            for        (int        i        =Convert.ToInt32(Math.Pow(2,
comboBox1.SelectedIndex)); i < listBox2.Items.Count; i++)
            {
                int vl_canal; //variable vl_canal como entero
                vl_canal                                                =
Convert.ToInt16(listBox2.Items[i].ToString().Substring(listBox2.Items[i].
ToString().Length - 2, 2)); //obtiene el número del canal de la trama E1
                MyE1.Canales[vl_canal].Text = "00000000";//asigna 00000000 al
canal seleccionado
                MyE1.Canales[vl_canal].BackColor  =  Color.White;  //asigna
blanco como color del canal de E1
            }
            MyE1.Width = MyE1.Width + Factor; //obtiene la longitud del E1
            Factor = Factor * (-1); //asigna 1 a factor
        }

    }
```

ANEXO B

Anexo B. Cálculos estadísticos del algoritmo de compresión.

A continuación se describe los cálculos estadísticos para las diferentes tecnologías y velocidades de transmisión.

Cálculos Estadísticos E1

Variables de intervienen en el cálculo:

Tiempo de Transferencia:

$$T\_trans = \frac{\#\ bits}{AB}$$

Se define como tiempo de transferencia a la relación entre el número de bits sobre el ancho de banda del canal.

BW	Máximo ancho de banda teórico del "enlace más lento" entre el host origen y el host objetivo (medido en bits por segundo).
P	Tasa de transferencia real en el momento de la transferencia (medida en bits por segundo)
T	Tiempo en el que se debe producir la transferencia de archivos (medido en segundos)
S	Tamaño del archivo en bits

Por ejemplo: Cuál es el tiempo que se necesita para transferir una información de 20 MB usando un ancho de banda de 128 Mbps?

20Mb = 20(1024) = 20480Kbytes
20480Kbytes = 20480(1024) = 20,971,520 bytes
20,971,520 bytes = 20,971,520(8) = 167,772,160 bits
128Mbps = 128(1000) = 128000kpbs
128000kpbs = 128,000(1000) = 128,000,000bps

Ahora sí, usando la formula:

T=(# bits)/(AB)=(167,772,160)/(128,000,000)…
T=1,31072 seg.

Nota: Este cálculo es el mismo para los cálculos de las otras velocidades.

Tiempo de propagación:

Distancia * Coeficiente retardo medio de TX

El tiempo de propagación está definido por el medio de transmisión.

Medio	Velocidad en [m/seg]
Vacío	$3.0 * 10^8$
Cobre	$2.3 * 10^8$
Fibra óptica	$2.0 * 10^8$

Tomando como referencia la ecuación de la velocidad **Vel = d/t**

Al despejar el tiempo se obtiene: **t = d/Vel**

Por ejemplo: Calcular el tiempo de propagación de una señal por un enlace de 10 Km utilizando fibra óptica

Entonces:

$$t = \frac{10\ Km}{2x10\ m/s}$$

De donde se tiene:

$$t = 10000 \times 2x10 \text{ seg}$$
$$t = 0{,}0002 \text{ seg}$$

Nota: Este cálculo es el mismo para los cálculos de las otras velocidades

Tiempo de procesamiento:

Una trama E1 dispone de dos canales que son reservados para control y sincronismo, de manera que estos no pueden llevar tráfico de datos. Por consiguiente el tiempo de procesamiento de cada trama corresponde a dos tiempos de byte por el número de tramas ocupadas.

$$T\_proc = \#\text{ tramas utilizadas} \times \text{tiempo de byte} \times \#\text{ canales de procesamiento}$$

Para conocer el tiempo de procesamiento de un cierto flujo de información se lo calcula considerando el número de tramas que dicha información ocupa para enviar los datos, también el tiempo de byte y el número de canales de procesamiento.

Para saber el número de tramas que necesitaría un cierto flujo de información se procede de la siguiente manera:

$$\#\text{ de tramas necesarias} = \frac{\#\text{ bytes (archivo)} \times 64\text{kbps}}{AB}$$

Por ejemplo: Se desea transmitir un archivo de 4 bytes (32 bits) por un canal de 128Kbps.

$$\#\text{ de tramas necesarias} = \frac{4\text{ bytes x }64\text{kbps}}{128\text{Kbps}}$$

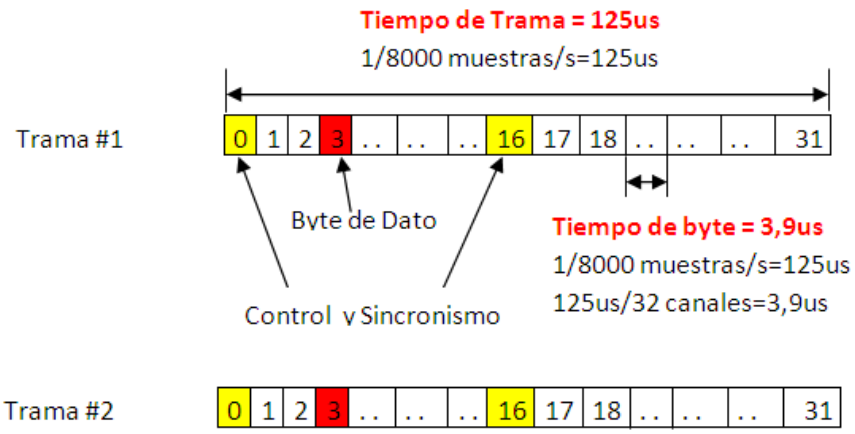
$$\#\text{ de tramas necesarias} = \frac{4\text{ bytes}}{2} = 2\text{ tramas}$$

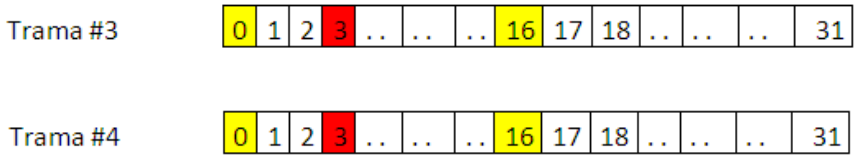
de tramas necesarias = 2 tramas

de tiempos de byte de procesamiento = 2 x # de tramas necesarias = 4

Para un archivo de tamaño 4 bytes ocupando un ancho de banda de 64 Kbps se tendría lo siguiente:

Cada trama E1 ocupará un time slot (canal) donde almacenará un byte del archivo, de manera que se necesitaría cuatro tramas E1 para transportar el archivo de 4 bytes.





Por cada trama E1 se procesan dos bytes de control (control y sincronismo) y para este caso se tiene un total de 8 bytes de control y cuatro de datos.

Para conocer el tiempo de byte del E1 se lo calcula de la siguiente manera:

$$T\_byte = 1seg/8000muestras/32\ canales$$
$$T\_byte = 0{,}00000390625\ seg$$

Con estos resultados se deduce que el Tiempo Total es el resultado de la suma entre los tiempos de transferencia, tiempo de propagación y tiempo de procesamiento.

$$Tiempo\_Total = Tiempo\_Transferencia + Tiempo\_Propagación + Tiempo\_Procesamiento$$

Cálculos Estadísticos STM-1

Variables de intervienen en el cálculo:

Tiempo de Transferencia:

$$T\_trans = \frac{\#\ bits}{AB}$$

Los mismos cálculos que para el caso anterior E1.

Tiempo de propagación: Distancia * Coeficiente retardo medio de TX

El tiempo de propagación está definido por el medio de transmisión.

Medio	Velocidad en [m/seg]
Vacío	$3.0 * 10^8$
Cobre	$2.3 * 10^8$
Fibra óptica	$2.0 * 10^8$

Tomando como referencia la ecuación de la velocidad **Vel = d/t**

Al despejar el tiempo se obtiene: **t = d/Vel**

Para calcular el tiempo de propagación se efectúa como en el caso anterior E1.

Tiempo de procesamiento:

Una trama STM-1 dispone de varias formas de demultiplexar a unidades más pequeñas (TU-12, AU3- AU-4)

Tiempo de procesamiento de trama TU-12.

La máxima cantidad de canales que se utilizan para transportar señales o información por un TU-12 son 30 canales de datos. Para alcanzar un TU-12 se parte de una señal de 2Mbps o E1, a este se le añaden bytes de bajo orden para poder ser mapeadas dentro de una señal de mayor capacidad. El tamaño de una trama TU-12 es de 36 bytes (30 para información útil y 6 de control)

Tal como en los cálculos de los E1, se debe conocer el número de tramas que se utilizará para transmitir los datos de una cierta información. De manera que se tiene lo siguiente:

$$\#\text{ de tramas necesarias} = \frac{\#\text{ de bytes por enviar / 30 canales}}{\text{Ancho de Banda / Canal mínimo}}$$

El valor del canal mínimo se lo considera 2048 Kbps.

Por ejemplo calcular el número de tramas necesarias para transportar 63 bytes por un canal de 4M utilizando dos tramas TU-12 concatenadas

$$\#\text{ de tramas necesarias} = \frac{63 / 30}{4096 / 2048}$$

.

$$\#\text{ de tramas necesarias} = \frac{2,1}{2}$$

de tramas necesarias = 1,05

de tramas necesarias = 2

La trama TU-12 dispone de seis canales que son reservados para control y sincronismo, de manera que estos no pueden llevar tráfico de datos. Por consiguiente el tiempo de procesamiento de cada trama corresponde a seis tiempos de byte por el número de tramas ocupadas.

Para el cálculo del tiempo de byte de un contenedor TU-12 se considera el mismo tiempo de byte de una trama E1, es decir 1 seg / 8000 muestras/s / 32 canales:

De esta manera se considera la fórmula para calcular el tiempo de procesamiento de un canal TU_12 de la siguiente forma:

T_proc = # tramas utilizadas x tiempo de byte x # canales de procesamiento

Para el caso anterior se calcula el tiempo de procesamiento:

$$T\_proc = \frac{63 / 30}{4096 / 2048} \times \frac{1 / 8000}{32} \times 6$$

T_proc = 46,8 us

Con estos resultados se deduce que el Tiempo Total es el resultado de la suma entre los tiempos de transferencia, tiempo de propagación y tiempo de procesamiento.

Tiempo_Total=Tiempo_Transferencia+Tiempo_Propagación+Tiempo_Procesamiento

Tiempo de procesamiento de AU-3.

Teóricamente, la velocidad de un canal VC-3 es de 44736 Mbps. Un AU-3 tiene una capacidad de bytes máxima de 774 bytes agrupados en 86 columnas de 9 filas (86x9), de los cuales 756 bytes son de información pura y 18 bytes de control.

Tal como en los cálculos de los E1, se debe conocer el número de tramas que se utilizará para transmitir los datos de una cierta información. De manera que se tiene lo siguiente:

$$\text{\# de tramas necesarias} = \frac{\text{\# de bytes por enviar / 756 bytes datos}}{\text{Ancho de Banda / Canal mínimo}}$$

El valor del canal mínimo se considera 44736 Mbps.

Por ejemplo calcular el número de tramas necesarias para transportar 63000 bytes por un canal de 45M utilizando un canal AU-3.

$$\text{\# de tramas necesarias} = \frac{63000 / 756}{44736000 / 44736000}$$

.

$$\text{\# de tramas necesarias} = \frac{83,33}{1}$$

\# de tramas necesarias = 83,33

\# de tramas necesarias = 84

Un canal AU-3 dispone de dieciocho canales que son reservados para control y sincronismo, de manera que estos no pueden llevar tráfico de datos. Por consiguiente el tiempo de procesamiento de cada trama corresponde a dieciocho tiempos de byte por el número de tramas ocupadas.

Para el cálculo del tiempo de byte de un contenedor AU-3 se considera lo siguiente: 1 seg / 8000 muestras/s / 774 canales:

De esta manera se considera la fórmula para calcular el tiempo de procesamiento de un canal AU-3 de la siguiente forma:

T_proc = # tramas utilizadas x tiempo de byte x # canales de procesamiento

Para el caso anterior se calcula el tiempo de procesamiento:

$$T\_proc = \frac{63000 / 756}{44736000 / 44736000} \times \frac{1 / 8000}{774} \times 18$$

T_proc = 244,18 us

Con estos resultados se deduce que el Tiempo Total es el resultado de la suma entre los tiempos de transferencia, tiempo de propagación y tiempo de procesamiento.

Tiempo_Total=Tiempo_Transferencia+Tiempo_Propagación+Tiempo_Procesamiento

Tiempo de procesamiento de AU-4.

Teóricamente, la velocidad de un canal VC-4 es de 139264 Mbps. Un AU-4 tiene una capacidad de bytes máxima de 2430 bytes agrupados en 270 columnas de 9 filas (270x9), de los cuales 2349 bytes son de información pura y 81 bytes de control.

Tal como en los cálculos de los E1, se debe conocer el número de tramas que se utilizará para transmitir los datos de una cierta información. De manera que se tiene lo siguiente:

$$\#\text{ de tramas necesarias} = \frac{\#\text{ de bytes por enviar / 2349 bytes datos}}{\text{Ancho de Banda / Canal mínimo}}$$

El valor del canal mínimo se considera 139264 Mbps.

Por ejemplo calcular el número de tramas necesarias para transportar 63000 bytes por un canal de 140M utilizando un canal AU-4.

$$\#\text{ de tramas necesarias} = \frac{63000 / 2349}{139264000 / 139264000}$$

.

$$\#\text{ de tramas necesarias} = \frac{26,81}{1}$$

de tramas necesarias = 26,81

de tramas necesarias = 27

Un canal AU-4 dispone de ochenta y un canales que son reservados para control y sincronismo, de manera que estos no pueden llevar tráfico de datos. Por consiguiente el tiempo de procesamiento de cada trama corresponde a dieciocho tiempos de byte por el número de tramas ocupadas.

Para el cálculo del tiempo de byte de un contenedor AU-4 se considera lo siguiente: 1 seg / 8000 muestras/s / 2430 canales:

De esta manera se considera la fórmula para calcular el tiempo de procesamiento de un canal AU-3 de la siguiente forma:

T_proc = # tramas utilizadas x tiempo de byte x # canales de procesamiento

Para el caso anterior se calcula el tiempo de procesamiento:

$$T\_proc = \frac{63000 / 2349}{139264000 / 139264000} \times \frac{1 / 8000}{2430} \times 81$$

T_proc = 112,5 us

Con estos resultados se deduce que el Tiempo Total es el resultado de la suma entre los tiempos de transferencia, tiempo de propagación y tiempo de procesamiento.

Tiempo_Total=Tiempo_Transferencia+Tiempo_Propagación+Tiempo_Procesamiento