Cabecera de Segmento TCP

Cabecera de formato fijo de 20 B + Opciones de cabecera + Datos opcional de 0 (acuses recibo y control) a Máx [65535 - 20 H\_IP- 20 H\_TCP=65495].

Campos Port Origen y Port Destino (16 bits): identifican puntos terminales locales de conexión,

Número de Secuencia (32): Posición del byte de Datos en este segmento.

Número de Reconocimiento (32):Acuse de recibo del siguiente byte esperado.

Long. de Cabecera TCP (4): En palabras de 32 bits por campo Opciones variable.

Campo Reservado sin Uso (6)

Banderas de un bit (6):

URG=1 si Apuntador Urgente (16) está en uso, indica offset en bytes, a partir del nº secuencia actual, donde comienzan Datos Urgentes.

ACK=1, indica que campo de reconocimiento es válido. Si=0, no contiene acuse recibo.

PSH=1, indica que este segmento requiere un PUSH (entregar datos a aplicación y no pasarlos al buffer)

RST=1, resetea la conexión.

SYN se usa para establecer conexiones. Solicitud conexión: SYN=1,ACK=0, Respuesta: SYN=1, ACK=1.

FIN=1, emisor no tien más datos para transmitir. Usado para liberar conexión.

Tamaño de Ventana Receptora (16):cant. Bytes que pueden enviarse, comenzando por el que se envía acuse recibo.

Suma de Comprobación (16):checksum de [Cabecera + Datos+ Pseudocabecera conceptual de Fig.6.14]  ®

 suma todas las palabras de 16 bits en Ca1, y luego obtiene el Ca1 de la suma.

El Rx hace el mismo cálculo con el segmento completo, incluído campo checksum, y debe dar cero.

Opciones (³0): permite agregar características extras no cubiertas en Header normal, como:

Gestión de Conexión TCP

Establecimiento de conexiones TCP emplean Protocolo de Acuerdo de 3 vías (Three-way Handshake).

Caso Normal:

Servidor espera pasivamente una conexión ejecutando primitivas LISTEN y ACCEPT. El Cliente ejecuta primitiva CONNECT especificando Dir. y Port IP con el que desea la conexión y tamaño máx. segmento TCP , ®Primitiva CONNECT envía segmento TCP con bit SYN=1 y ACK=0, y espera respuesta.

Si proceso escucha en el port, recibe segmento TCP entrante y puede aceptar (ó rechazar conexión), devolviendo segmento con acuse recibo.

Colisión llamadas  sólo se establecerá una conexión, no dos, identificada por (x,y)

Para Liberar Conexiones, cualquiera de las partes envía segmento TCP con bit FIN seteado, y al reconocer el FIN, ese sentido se apaga. Cuando ambos sentidos se apagan, se libera conexión.

Control de Congestión con TCP

Congestión: cuando carga ofrecida a red es mayor que la que puede manejar

Capa de Red intenta manejarla, pero solución real es bajar la Tasa de Datos , trabajo para TCP.

Mayoría de terminaciones de Temporización (Timeout) en Internet es por congestión en enrutadores que por errores de tx por ruidos en líneas.

Solución de Internet a 2 problemas potenciales de Capacidad del Rx y capacidad de la red, es que Tx mantiene,

             Ventana = mín { Ventana anunciada por Rx ;  Ventana de Congestión }

Vent\_Congestión / si acuse recibo segmento OK ® Vent\_Cong se duplica,..., y crece exponencialmente hasta alcanzar 1/2 Tamaño Vent\_Receptora, en que reduce razón del incremento, pasando a ser lineal (+1): Algoritmo de Arranque Lento  (Jacobson,1988), en implementaciones de TCP.

Si ocurre un Timeout, se establece el Umbral (inicial de 64kB) en la 1/2.Vent\_Cong. Actual y la Vent\_Cong se restablece a 1 kB.      Luego usa arranque lento para determinar lo que puede manejar la red, hasta alcanzar el umbral.

Algoritmo de Retransmisión Adaptable en TCP

Temporizador de Retransmisión se inicia al enviar segmento, y si termina antes que llegue acuse recibo, se retransmite el segmento y reinicia temporizador.

Entorno de TCP distinto al de protocolos de Enlace de datos ® funciones de densidad de probabilidad del tiempo que tarda en regresar acuse de recibo

Algoritmo Dinámico (Jacobson,1988): para c/conexión TCP mantiene una variable RTT (Round Trip Time): estimación del Tiempo de Ida y Vuelta al destino en cuestión y mide M: tiempo que tardó acuse recibo (nueva muestra de RTT),® actualiza:

*RTT = a.RTT + (1-a).M  ,  con a=7/8 :  Factor Amortiguamiento.*

ÞTimeout= b.RTT , normalmente, con b=2 inicialmente, pero experiencia demuestra que era inflexible y no respondía cuando existen variaciones.

® Se propone b proporcional a Desviación Media  de Función y mantener otra variable Desviación

 D = a.D +(1-a).|RTT-M| Þ        Timeout= RTT + 4.D, usado en mayoría de implementaciones TCP.

Algoritmo de Karn: cuando termina de temporizar y retransmite segmento,®   no actualizar el RTT con ninguno de los segmentos retransmitidos por confusión en los acuses de recibo, sino ®  Timeout=Timeout\_viejo. 2, hasta que los segmentos pasan a la primera vez.