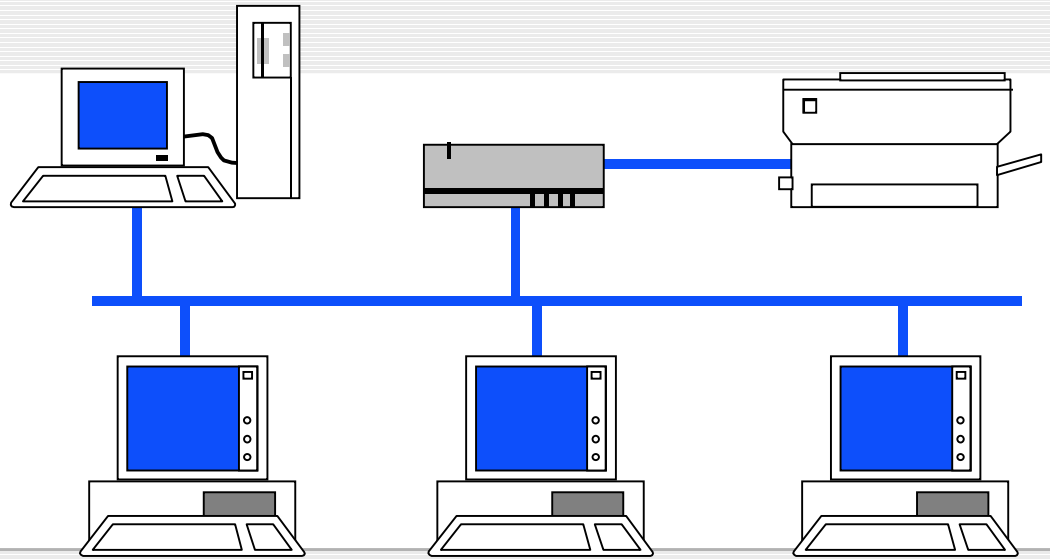
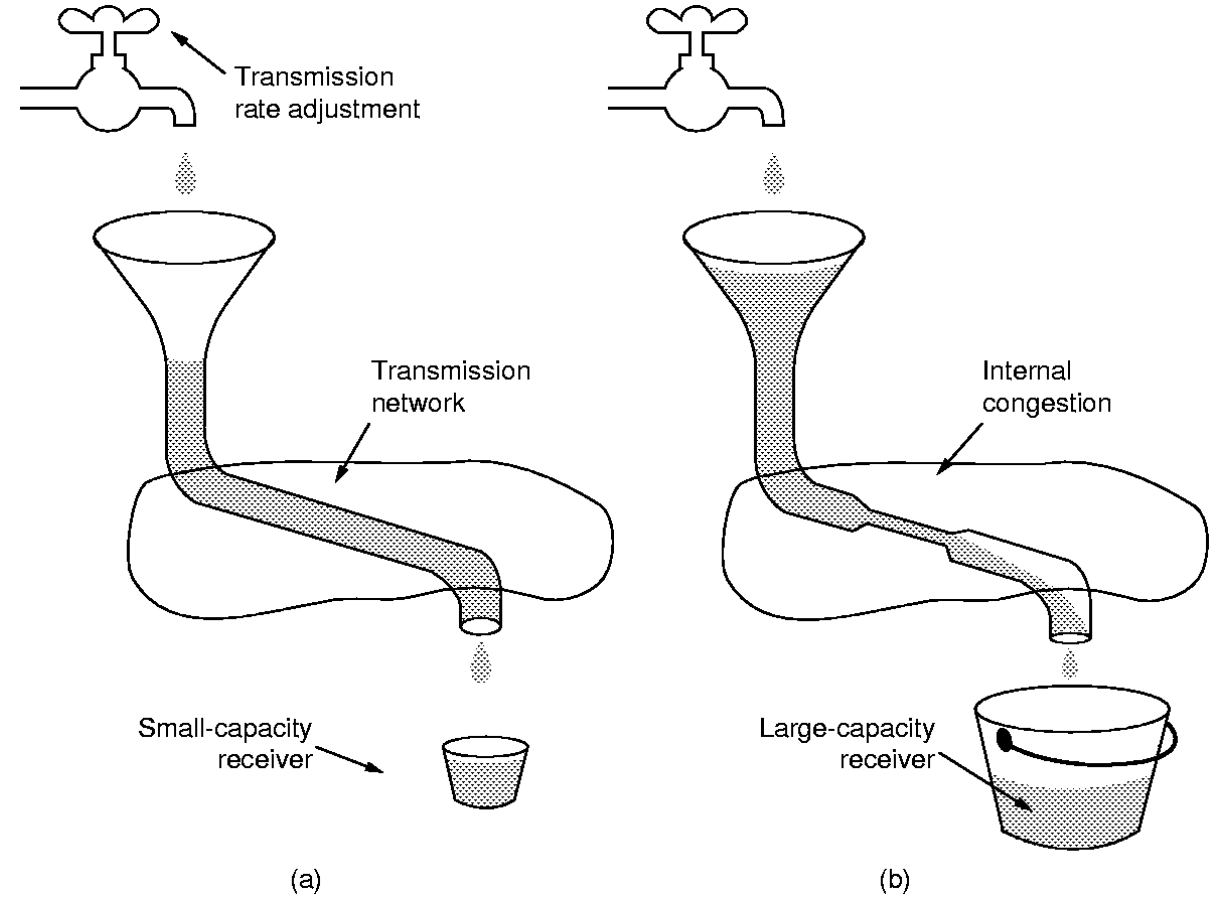
Redes de Computadores

Capa de Transporte
TCP y UDP



Control de Congestión



- 2 posibles overflows (que causarán descarte de paquetes)
  - (a) el RX no es capaz de recibir a la tasa que el TX le envía
  - (b) existe congestión, por lo que existe overflow en algún router intermedio

Control de Congestión

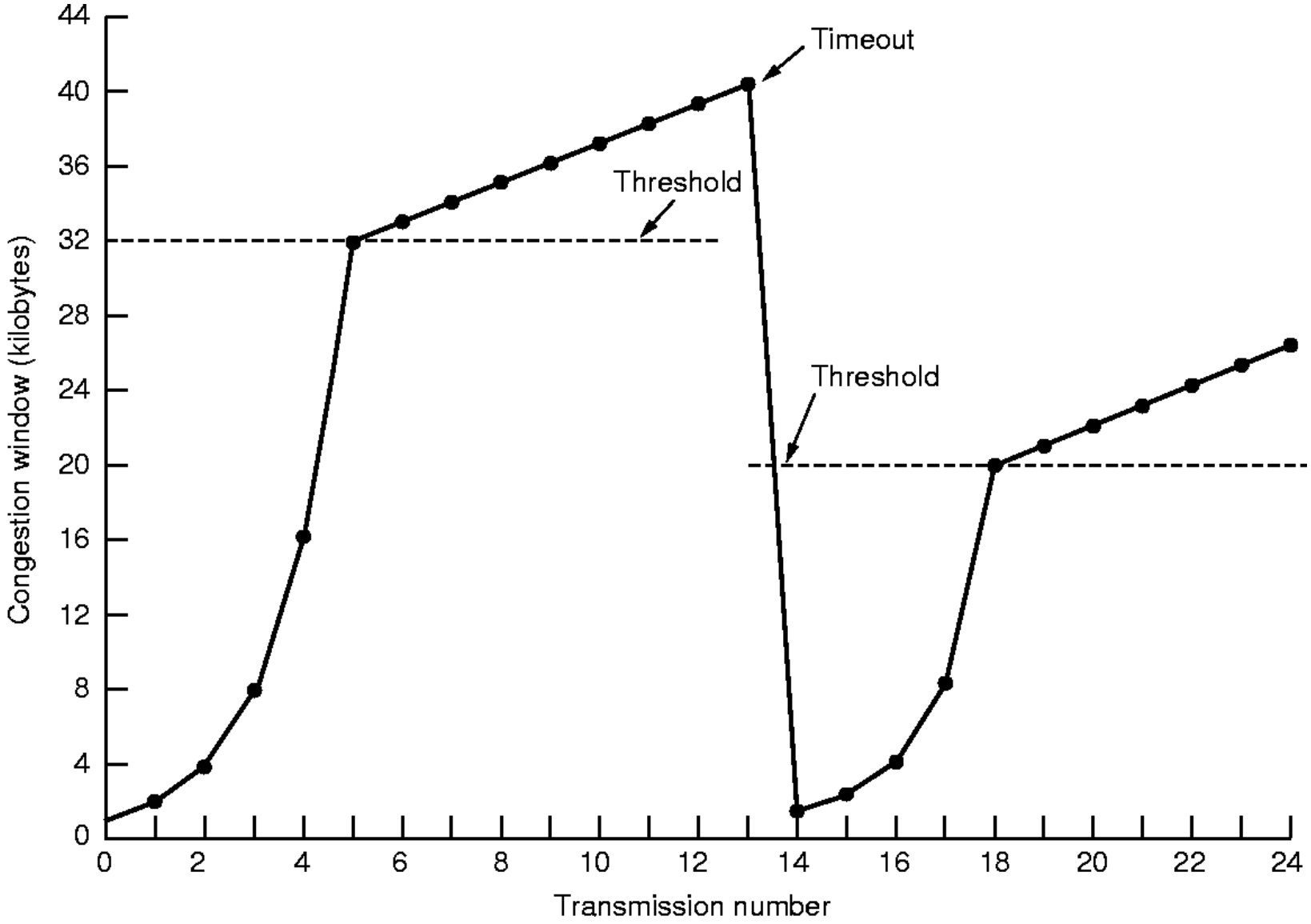
- TCP implementa 2 ventanas de TX en cada extremo:
  - la que refleje la capacidad del receptor
    - Control de Flujo (Window size) visto anteriormente
  - la que refleje la capacidad de la red
    - Control de Congestión (ventana de congestión)
- Ambas indican la cantidad de bytes que puede enviar un TX. (al transmitir, se escoge el tamaño mínimo de las ventanas)
- Cuando se inicializa una conexión, la ventana de congestión equivale a un segmento TCP.
- Si se recibe el ACK del segmento dentro del TIMEOUT, significa que la "red" no está congestionada, por lo que incrementa el tamaño de la ventana de congestión a 2 segmentos
- Si ambos segmentos son ACK, entonces la ventana crece a 4 segmentos...etc.

Control de Congestión

- La ventana crece exponencialmente hasta que exista un TIMEOUT o se alcance la ventana del receptor (control de flujo)
- Este algoritmo se conoce como “Slow Start” o “Partida Lenta”
- Existe un 3er parámetro, el umbral o “threshold”, inicialmente de 64 Kbyte
- Cuando ocurre un TIMEOUT, este umbral es seteado a la mitad del tamaño de la ventana de congestión actual, y la ventana de congestión es seteada a 1 segmento
- Empieza a trabajar el sistema “Slow Start” en forma exponencial hasta llegar al umbral, luego la ventana de congestión crece en forma lineal de a 1 segmento.

Control de Congestión

umbral ya había sido bajado a 32Kbyte



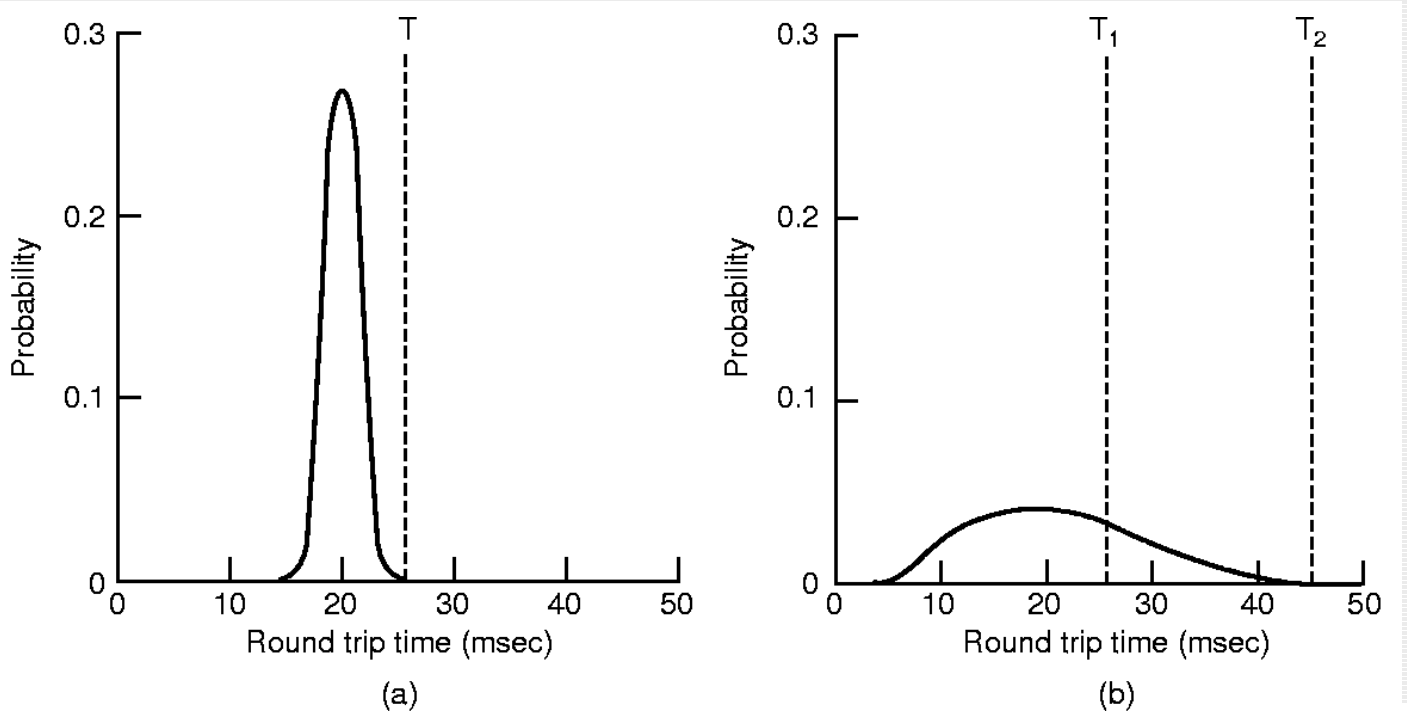
Timers (Temporizadores)

- TCP ocupa 4 Timers (o timeouts)

- **1.- Timeout de Re-transmisión**
  - Cuando un segmento es emitido, este timer empieza a contar
  - si se recibe un ACK de este segmento antes que este timer expire, el timer es detenido
  - si no se recibe un ACK antes que el timer expire, el segmento es re-TX (y el timer empieza a contar nuevamente)
  - ¿ en cuánto debe setearse este timer ?

Timers (Temporizadores)

- Tiempo de ida y vuelta (o de recepción de ACK)
  - (a) en Capa 2
  - (b) en TCP
- setear como timeout a T1 causa muchas re-TX
- setear como timeout a T2 causa bajo rendimiento



Timers (Temporizadores)

- Para calcular el Timeout de re-TX óptimo, éste debe ser calculado cada vez que llega un ACK (Jacobson 1988)
- Si un ACK llega antes de tiempo, M[s], entonces se calcula la variable RTT (Round Trip Time):
  - $RTT = \alpha\ RTT + (1-\alpha)\ M$ con $\alpha=7/8$
- entonces el Timeout = β RTT con $\beta = 2$.

Timers (Temporizadores)

- Resulta que al mantener β constante, tampoco satisface en situaciones donde la varianza crece
- Entonces se lleva registro de otra variable, D, la desviación
  - $D = \delta D + (1- \delta) |RTT-M|$
- (NOTA: D no es la desviación estándar, sino una aproximación)
- Ahora el Timeout = RTT + 4 D

Timers (Temporizadores)

2.- Timeout de Persistencia

- diseñado para prevenir “deadlock”
- Situación:
  - un receptor avisa que posee ventana 0 y el transmisor detiene el envío
  - después de un tiempo, el RX puede recibir más información y envía un mensaje indicando el nuevo tamaño de ventana
  - este mensaje se pierde
  - ambos se quedan en deadlock

  - Si este Timer expira, el TX envía un mensaje consultando la ventana de recepción
  - Si aún es cero, este Timer se re-inicializa y se repite el ciclo
  - Si no es cero, el TX empieza a enviar nuevos segmentos.

Timers (Temporizadores)

3.- Keepalive Timer

- si una conexión ha estado ociosa por un tiempo, este Timer expira y causa que una de las partes envíe un mensaje al otro extremo para saber si aún está ahí
- Si no hay respuesta, la conexión es cerrada

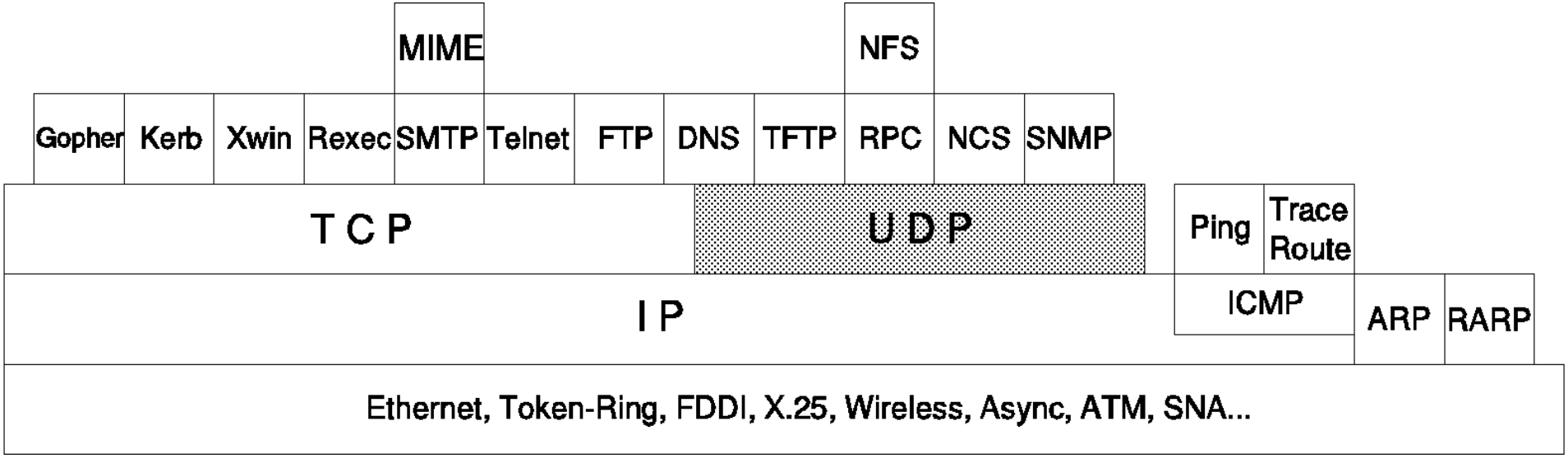
4.- TIMED WAIT

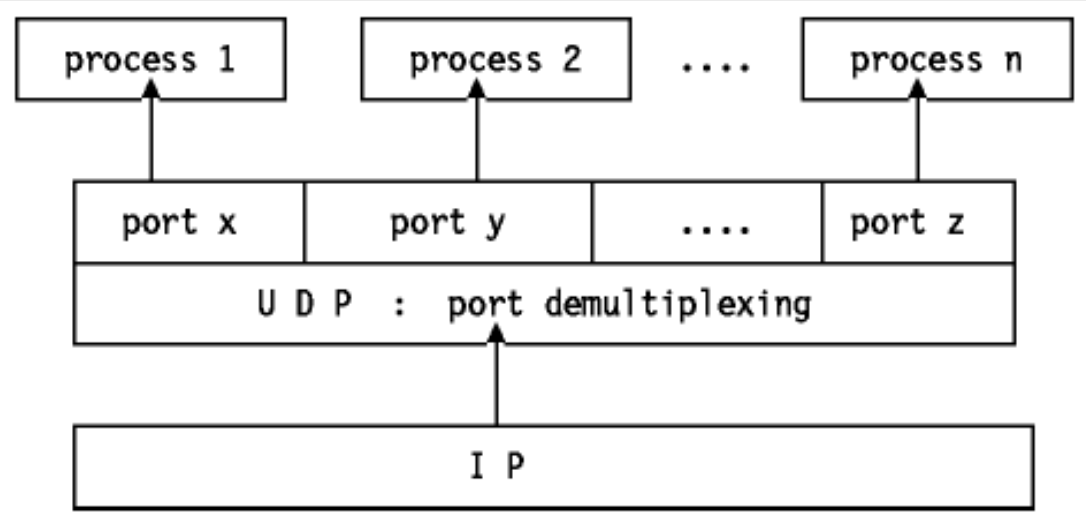
- Este timer se ocupa para asegurar de que antes de cerrar una conexión, se hayan recibido todos los mensajes que aún circulan por la red
- equivale a 2 tiempos de vida de un paquete (2 RTT ?)

UDP (RFC 768)

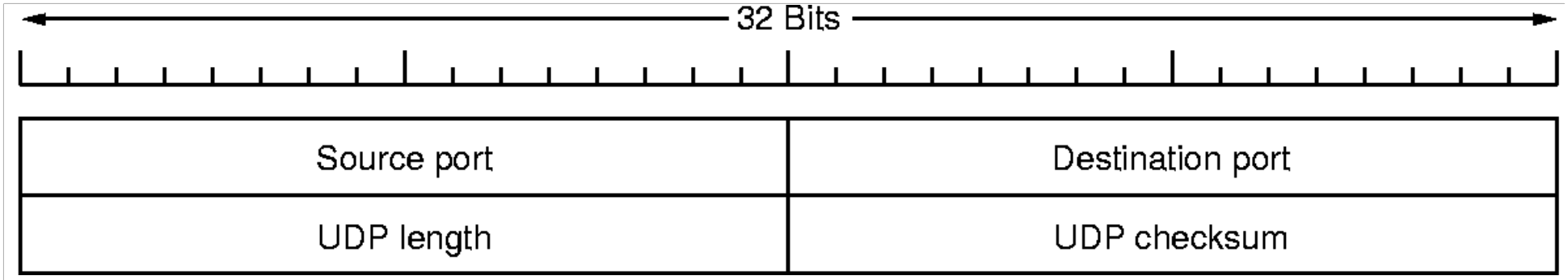
- User Datagram Protocol
  - transporte de datos sin conexión
  - permite a las aplicaciones implementar una comunicación tipo consulta-respuesta en forma rápida, sin tener que establecer una conexión
  - se considera como una interface entre capa aplicación e IP
  - sin corrección de errores (sólo detección)
  - sin control de flujo
  - tiempo de setup (retardo) bajo comparado con TCP
  - Aplicaciones
    - Base de Datos
    - NFS, SNMP, DNS, TFTP
    - otros..

UDP


MIME
NFS
Gopher
Kerb
Xwin
Rexec
SMTP
Telnet
FTP
DNS
TFTP
RPC
NCS
SNMP
T C P
U D P
Ping
Trace Route
I P
ICMP
ARP
RARP
Ethernet, Token-Ring, FDDI, X.25, Wireless, Async, ATM, SNA...


process 1
process 2
....
process n
port x
port y
....
port z
U D P : port demultiplexing
I P

UDP



- Puerto Origen y Destino
- Length: largo del Header (8bytes) + Data
- Checksum: permite detección de errores.