Protocolos de transporte

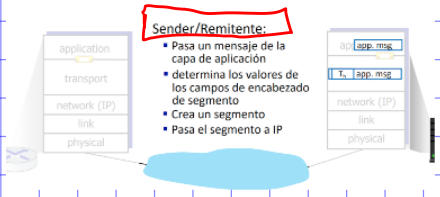
TCP

UDP

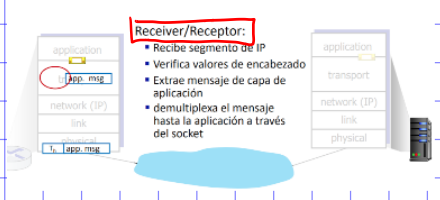
Capa de transporte → Comunicación entre procesos

Capa de red → Comunicación lógica entre hosts

Acciones de la capa de transporte



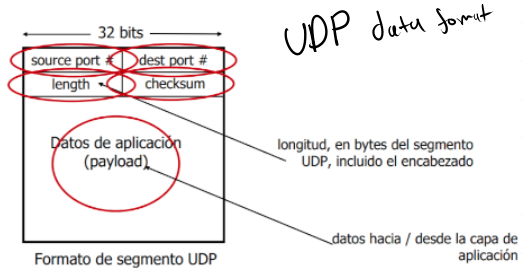
Acciones de la capa de transporte



En resumen

- Multiplexación, demultiplexación: basado en segmentos, valores de campo de encabezado de datagrama
- **UDP:** demultiplexación utilizando el número de puerto de destino (solamente)
- **TCP:** demultiplexación utilizando 4 tuplas: direcciones IP de origen y destino, y números de puerto

UDP data format



Formato de segmento UDP

ejemplo: suma dos enteros de 16-bit

	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
wraparound	(1)	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
sum		1	0	1	1	1	0	1	1	1	0	1	1	1	1	0	0
checksum		0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Nota: al sumar números, se debe agregar un arrastre del bit más significativo al resultado

① RDT → reliable data transfer protocol.

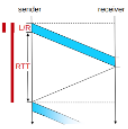
Desempeño de rdt3.0 (stop-and-wait)

- U_{sender}: *utilization* – Fracción de tiempo remitente ocupada enviando
- Ejemplo: 1 Gbps link, 15 ms prop. delay, 8000 bit packet
  - tiempo para transmitir el paquete al canal:

$$D_{trans} = \frac{L}{R} = \frac{8000\ bits}{10^9\ bits/sec} = 8\ microsecs$$

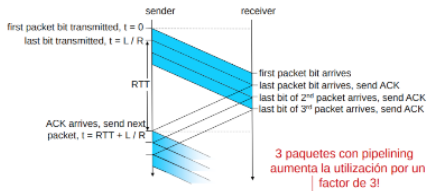
rdt3.0: operación stop-and-wait

$$U_{sender} = \frac{L/R}{RTT + L/R} = \frac{.008}{30.008} = 0.00027$$



- rdt 3.0 protocolo (el rendimiento no es el mejor)
- El protocolo limita el rendimiento de la infraestructura subyacente (enlace)

Pipelining: utilización incrementada



3 paquetes con pipelining aumenta la utilización por un factor de 3!

$$U_{sender} = \frac{3L/R}{RTT + L/R} = \frac{.0024}{30.008} = 0.00081$$

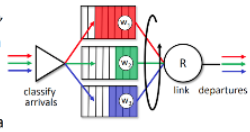
Reglas de Scheduling: weighted fair queueing

Weighted Fair Queuing (WFQ):

- Round Robin generalizado
- Cada clase, *i*, tiene un peso, w_i, y recibe una cantidad ponderada de servicio en cada ciclo:

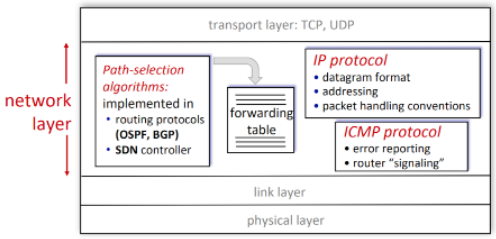
$$\frac{W_i}{\sum_j W_j}$$

- garantiza un ancho de banda mínimo (por clase de tráfico)

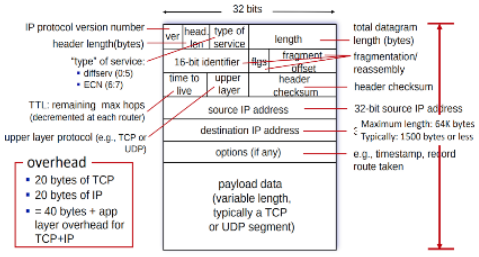


Capa de red: Internet

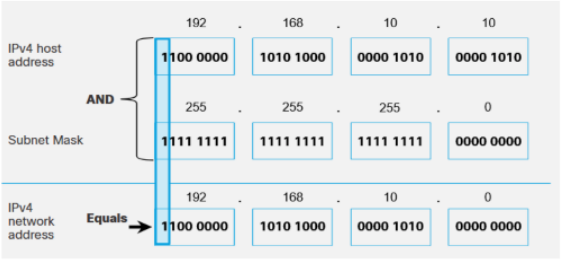
Host y router: funciones en la capa de red



IP Datagram (formato)



IP addressing: CIDR



Direcciones IP:

1. ¿Cómo obtiene un host la dirección IP dentro de su red (parte del host de la dirección)?
2. ¿Cómo obtiene una red una dirección IP por sí misma (parte de la dirección de red)?

¿Cómo obtiene el host la dirección IP?

- Codificada por sysadmin en archivo de configuración(e.g., /etc/rc.config in UNIX)
- DHCP: Dynamic Host Configuration Protocol: obtener la dirección de forma dinámica desde un servidor "plug-and-play"

IP fragmentation/reassembly

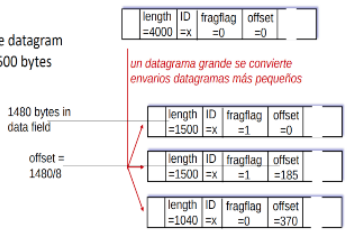
Ejemplo:

- Un datagrama de 4000 bytes llega al enrutador y se debe enviar a través de un enlace con MTU de 1500 bytes
- 20 bytes de encabezado, 3980 bytes de payload (datos)
- Se deben armar 3 paquetes (datagramas IP)
  - Paquete 1: 1480 bytes (byte 0 al 1479 - parte 0 a 184), offset: 0
  - Paquete 2: 1480 bytes (byte 1480 al 2959 - parte 185 a 369), offset: 185
  - Paquete 3: 1480 bytes (byte 2959 al 3979 - parte 370 a 498), offset: 370

IP fragmentation/reassembly

Ejemplo:

- 4000 byte datagram
- MTU = 1500 bytes



IP fragmentation/reassembly

Ejemplo:

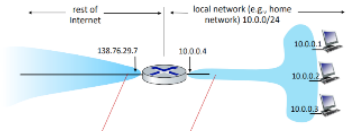
- 4000 byte datagram
- MTU = 1,500 bytes

	length =4000	ID =x	fragflag =0	offset =0	
	=1040	=x	=0	=370	

Fragment	Bytes	ID	Offset	Flag
1st fragment	1,480 bytes in the data field of the IP datagram	identification = 777	offset = 0 (meaning the data should be inserted beginning at byte 0)	flag = 1 (meaning there is more)
2nd fragment	1,480 bytes of data	identification = 777	offset = 185 (meaning the data should be inserted beginning at byte 1,480. Note that 185 · 8 = 1,480)	flag = 1 (meaning there is more)
3rd fragment	1,020 bytes (= 3,980–1,480–1,480) of data	identification = 777	offset = 370 (meaning the data should be inserted beginning at byte 2,960. Note that 370 · 8 = 2,960)	flag = 0 (meaning this is the last fragment)

NAT: network address translation

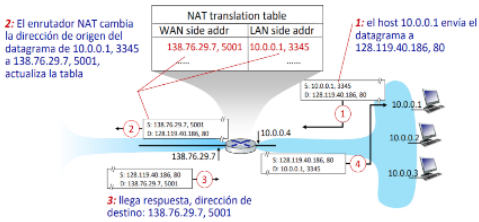
NAT: todos los dispositivos en la red local comparten solo una dirección IPv4 en lo que respecta al mundo exterior



todos los datagramas *saliendo* red local tienen *la misma* dirección NAT IP de origen: 138.76.29.7, pero diferentes puertos de origen

Los datagramas con origen o destino en esta red tienen una dirección 10.0.0/24 para el origen.

NAT: network address translation



OpenFlow: ejemplos

Destination-based forwarding:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	51.6.0.8	*	*	*	*	port6

Los datagramas IP destinados a la dirección IP 51.6.0.8 deben enviarse al puerto de salida 6 del enrutador

Firewall:

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	*	*	*	*	*	22	drop

Bloquear (no reenviar) todos los datagramas destinados al puerto TCP 22(ssh port #)

Switch Port	MAC src	MAC dst	Eth type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Prot	IP ToS	TCP s-port	TCP d-port	Action
*	*	*	*	*	*	128.119.1.1	*	*	*	*	*	drop

Bloquear (no reenviar) todos los datagramas enviados por el host 128.119.1.1

OpenFlow

- match+action: unifica diferentes tipos de dispositivos

Router
- *match:* prefijo de IP de destino más largo
- *action:* reenviar a capa de enlace

Switch
- *match:* dirección MAC de destino
- *action:* reenviar o permitir flujo

Firewall
- *match:* Direcciones IP y números de puerto TCP / UDP
- *action:* permite o bloquea

NAT
- *match:* Direcciones IP y números de puerto
- *action:* reescribir la dirección y el puerto