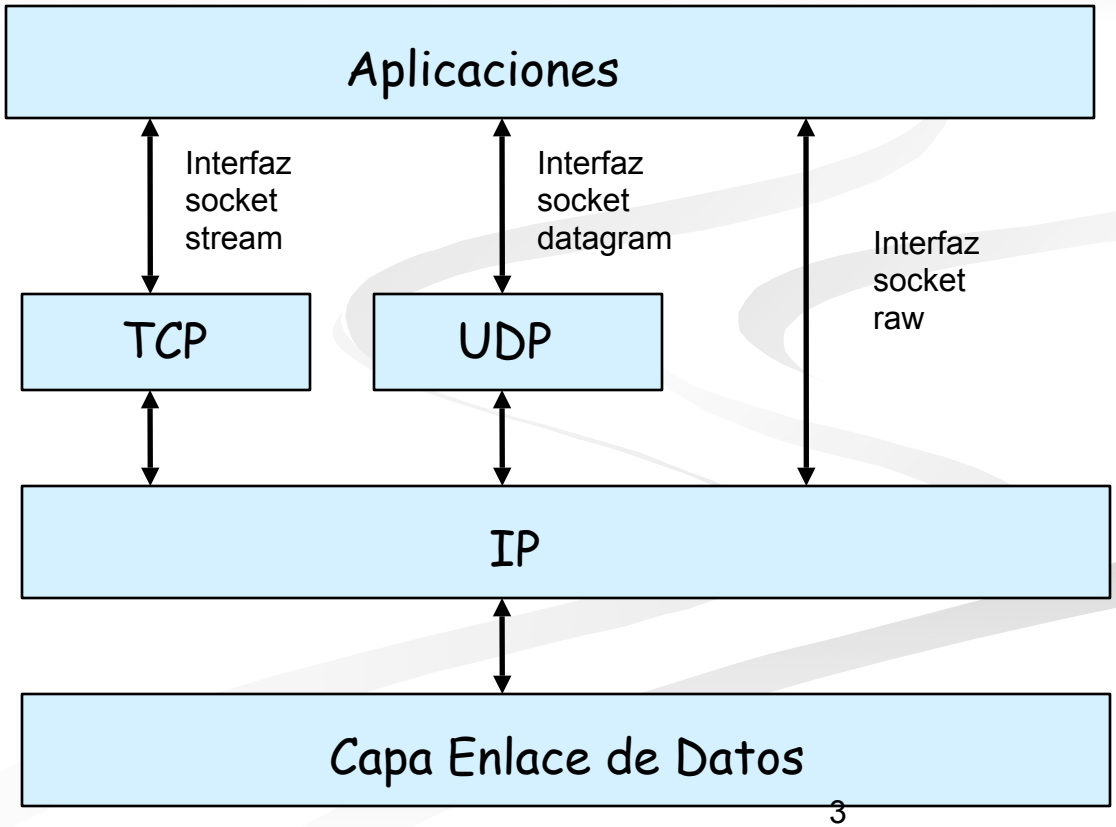
Raw Sockets & Data Link Access

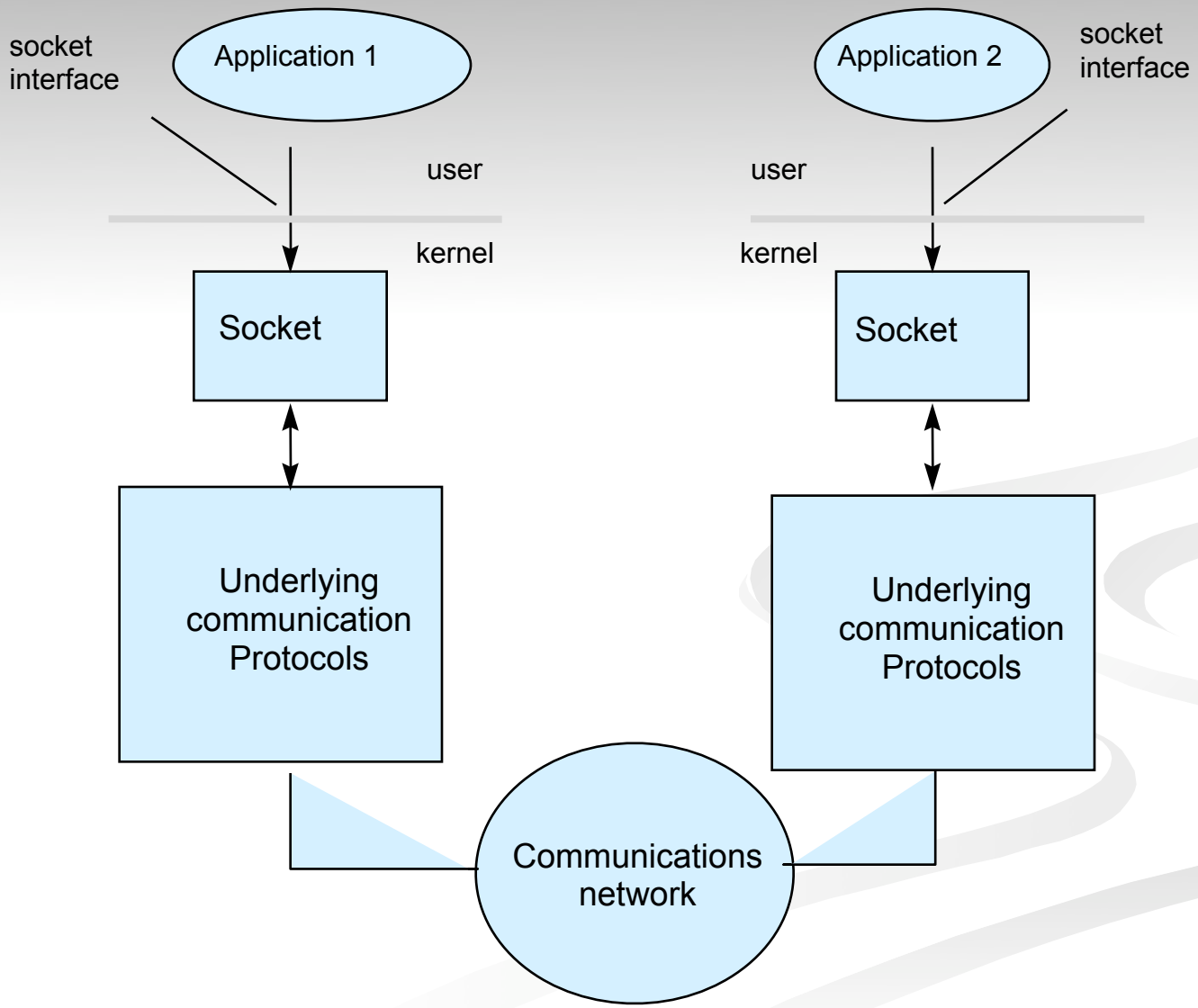
Session Overview

- Interfacing at
  - Network Layer
    - Sample Applications
  - Data link layer
    - Sample Applications
- Generic Packet Capture library

API de sockets

- Introducido en 1981 por Unix BSD 4.1
  - implementado como system calls
  - originalmente para Unix, pero Windows y Linux son parecidos
- Tipos de servicio
  - datagrama (UPD)
  - stream (TCP)
  - raw (IP)





Copyright ©2000 The McGraw Hill Companies

Raw Sockets

- Proporcionan tres características no ofrecidas normalmente por los sockets TCP y UDP
  - Dejan que el programador lea y escriba mensajes ICMP (Internet Control Message Protocol) e IGMP (Internet Group Message Protocol).
  - Aplicaciones pueden leer y escribir datagramas IPv4 sin que el núcleo procese un campo del protocolo IPv4
    - por ejemplo: OSPF con el campo de protocolo con un valor de 89.
  - Un proceso puede construir su propio encabezado IPv4, usando la opción IP_HDRINCL de la llamada setsockopt( )

Usando sockets tipo RAW

- Para usar sockets de tipo raw en Unix es necesario contar con privilegios de root
- Para crear un socket tipo raw es necesario usar la llamada socket() con los siguientes parámetros

```
s=socket( AF_INET, SOCK_RAW, [protocolo])
```

- A partir de esto es posible enviar/recibir sobre este socket.
- Sockets tipo raw pueden usarse para generar/recibir paquetes de un tipo que el kernel no soporta explicitamente.

Valores protocolos

- Valores usados para definir el campo de Protocolo en el encabezado IP
  - IP (dummy) IPPROTO_IP 0
  - ICMP IPPROTO_ICMP 1
  - IGMP IPPROTO_IGMP 2
  - Gateway IPPROTO_GGP 3
  - TCP IPPROTO_TCP 6
  - PUP IPPROTO_PUP 12
  - UDP IPPROTO_UDP 17
  - XND IDP IPPROTO_IDP 22
  - Net Disk IPPROTO_ND 77
  - Raw IP IPPROTO_RAW 255

Opciones llamada setsockopt()

- Usado para modificar opciones sockets
  - `IP_ADD_MEMBERSHIP`
    - Join a multicast group on a specified local interface
  - `IP_DROP_MEMBERSHIP`
    - Leave a multicast group
  - `IP_MULTICAST_IF`
    - Specify the interface for *outgoing* multicast datagrams sent on this socket
  - `IP_MULTICAST_TTL`
    - Set the IPv4 TTL parameter (if not specified, default=1)
  - `IP_MULTICAST_LOOP`
    - Enable or disable local loopback (default is enabled)

La opción IP_HDRINCL de setsockopt( )

- Si la opción `IP_HDRINCL` NO esta activada
  - el núcleo construye el encabezado de IP y pone los datos del proceso después del encabezado
  - él núcleo asigna el campo de protocolo del encabezado IPv4
- Si la opción `IP_HDRINCL` esta activada
  - el proceso construye todo el encabezado IP, excepto el checksum del encabezado IPv4 (siempre calculado por el núcleo) y el campo de identificación IPv4 (si se le asigna un valor de cero, el núcleo lo asigna).
- El núcleo fragmenta paquetes raw que excedan la interfaz de salida MTU

Raw Sockets: socket input

- Paquetes TCP y UDP nunca son entregados a un raw socket
- La mayoría de los paquetes ICMP y todos los paquetes IGMP se pasan a un raw socket
- Si el núcleo no entiende el campo de protocolo de un datagrama IP, se pasa a un raw socket
- Si el datagrama de entrada esta fragmentado, no se pasa nada al raw socket hasta que el reensamblado sea hecho.
- Núcleo examina todos los raw sockets de todos los procesos para decidir a quien se le entregan los datagramas que lleguen.

Internet Control Message Protocol

- ICMP definido en el RFC 792
- Mensajes ICMP
  - Interroga a nodo(s) por información
  - Reporta condiciones de error
- Mensajes ICMP son transportados como datagramas IP
  - ICMP usa o esta debajo de IP
- Mensajes ICMP son procesados por IP, UDP, o TCP
  - IP, TCP, y UDP "usan" o están abajo de ICMP

Raw sockets

- Raw sockets bypass the transport layer and directly communicate with layer 3.
- Using raw sockets, a user process can write the layer 3 header instead of the kernel doing it.
- It gives more flexibility to programmers as they access the layer 3 directly.

Creating raw sockets

```
socket(AF_INET, SOCK_RAW, protocol)
```

protocol is IPPROTO_xxx, defined in <netinet/in.h>

- Binding in raw sockets
  - A raw socket is bound to a particular interface on the host as there is no concept of port numbers.
  - Source address of the outgoing packets is the address of the bound interface.

Creating raw sockets…

- Connection management in raw sockets
  - There is no concept of connection management as such.
  - Connect call can be used on a raw socket which binds the raw socket to a specific destination address.
  - This allows us to use specific I/O calls such as read,write,send and recv which do not carry explicit address information.

Output using raw sockets

- There is no difference in the syntax of I/O calls for raw sockets.
  - Normal I/O : `sendto()` or `sendmsg()`
  - `write()`, `writev()`, `send()` can be used if socket is connected.
- Kernel fragments the raw packets that exceed the MTU

Output using raw sockets…

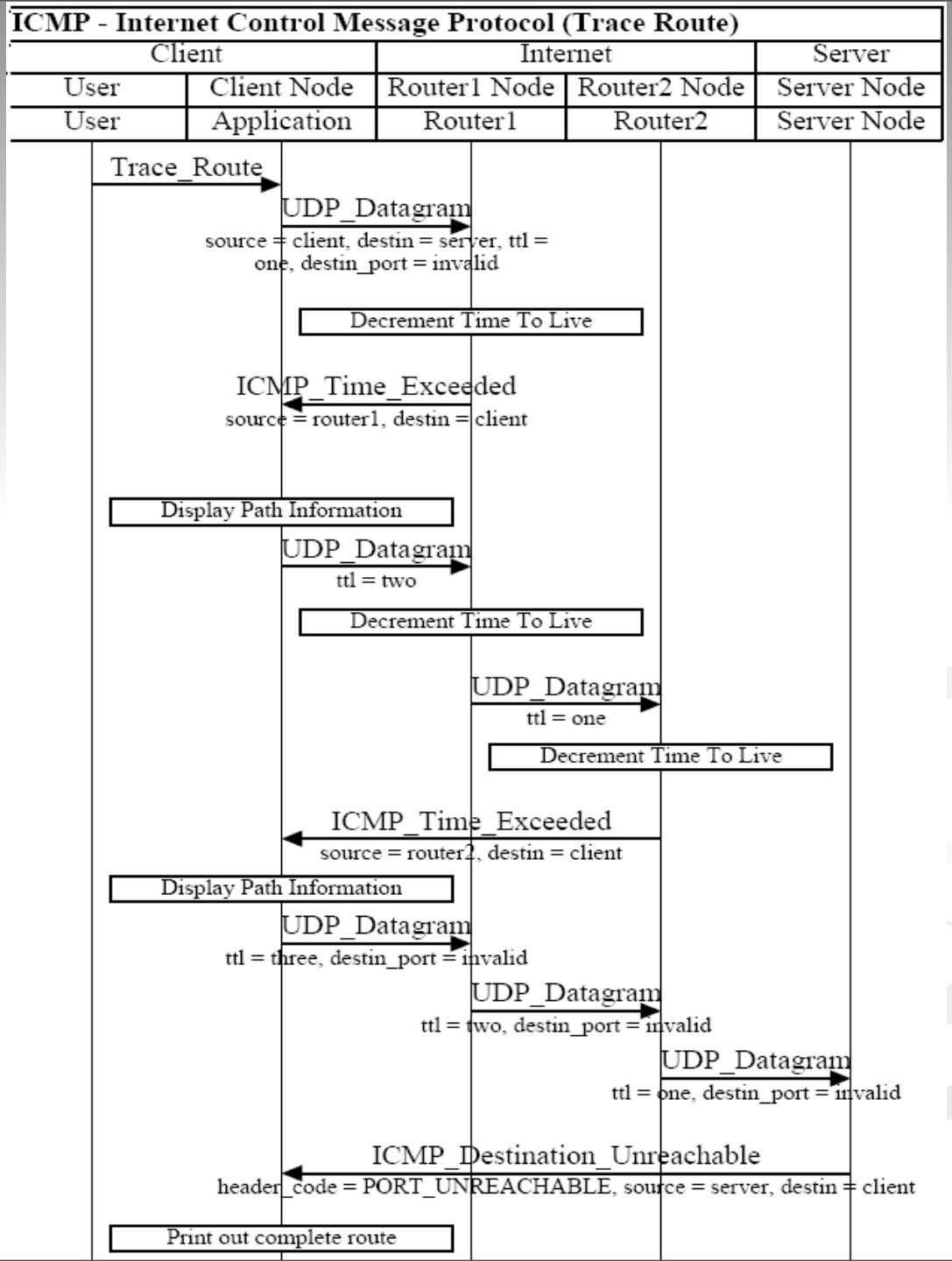
- `setsockopt()` to set IP options (optional header in the IP datagram)
- `IP_HDRINCL` option for generating IP header
- **`setsockopt (sockfd,IPPROTO_IP,IP_HDRINCL,&on,sizeof(on))`**
  - Excludes IP header checksum
  - Optionally Excludes IP identification field
- Other IP options not processed by kernel if `IP_HDRINCL` is included

Processing raw sockets

- UDP/TCP packets never passed to raw sockets
- Most ICMP packets are passed
- All IGMP packets are passed
- If IP protocol field is not understood, pass it to raw socket
- Reassemble datagrams before passing to raw sockets.

Processing raw sockets

- Matching datagrams are delivered
  - Protocol field in the IP header
  - Source IP in the IP header
  - Destn. IP in the IP header
- Deliver all datagrams if raw socket created with protocol value of 0
- If more matches are found, copies of datagrams passed to each matching raw socket
- If IP datagrams are fragmented, datagrams are delivered to application after reassembly


ICMP - Internet Control Message Protocol (Trace Route)
Client
Internet
Server
User
Client Node
Router1 Node
Router2 Node
Server Node
User
Application
Router1
Router2
Server Node
Trace_Route
UDP_Datagram
source = client, destin = server, ttl = one, destin_port = invalid
Decrement Time To Live
ICMP_Time_Exceeded
source = router1, destin = client
Display Path Information
UDP_Datagram
ttl = two
Decrement Time To Live
UDP_Datagram
ttl = one
Decrement Time To Live
ICMP_Time_Exceeded
source = router2, destin = client
Display Path Information
UDP_Datagram
ttl = three, destin_port = invalid
UDP_Datagram
ttl = two, destin_port = invalid
UDP_Datagram
ttl = one, destin_port = invalid
ICMP_Destination_Unreachable
header_code = PORT_UNREACHABLE, source = server, destin = client
Print out complete route

Traceroute Program