



# Host Networking

Jeffrey Shafer



# Routing on the Host

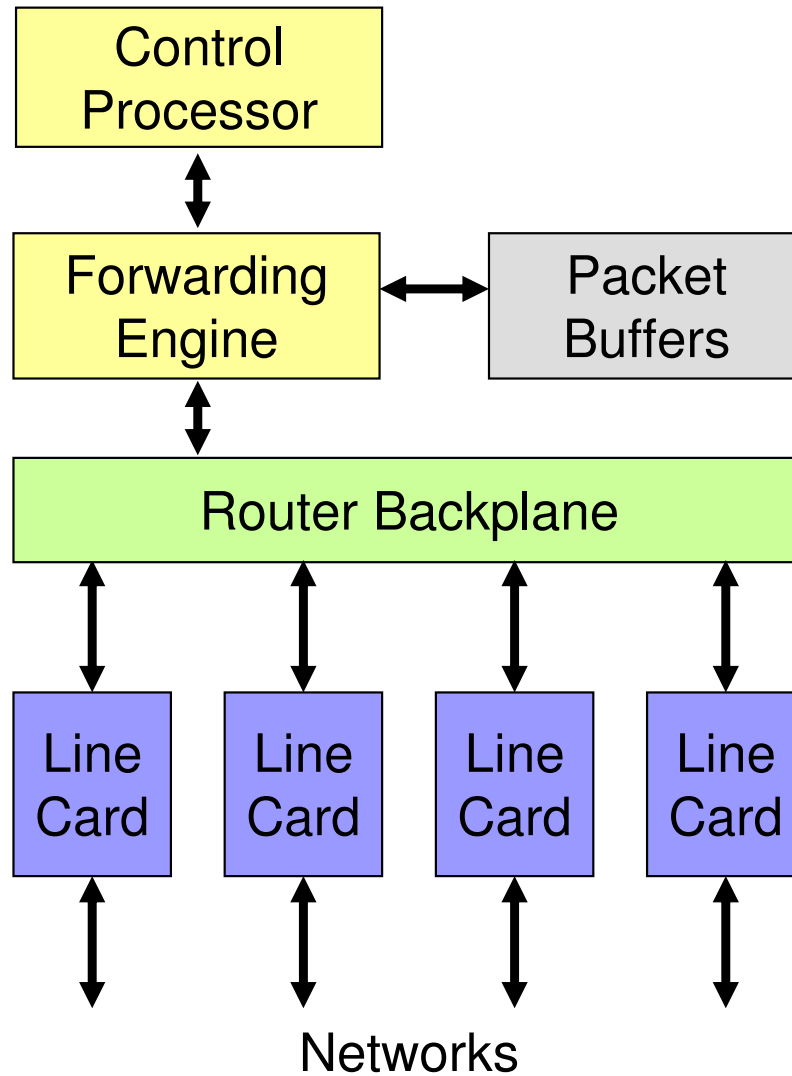
## ■ Router functionality

- Receive packets
- Determine next hop IP and port
- Find MAC address for next hop
- Update packet
- Forward packet to correct port

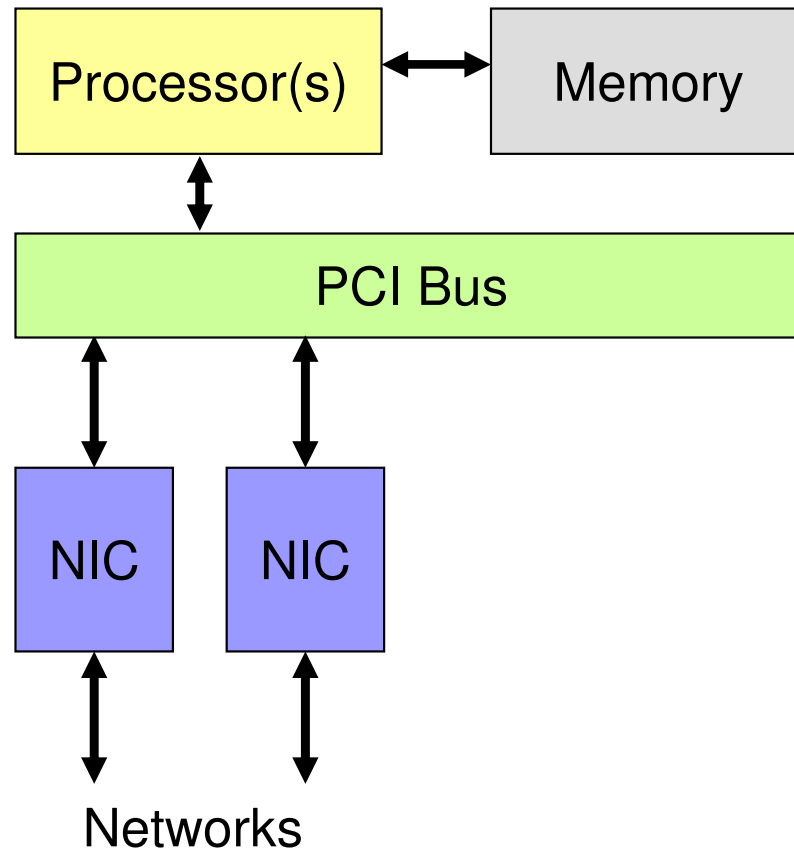
## ■ Host functionality

- ???

# Router



# Host





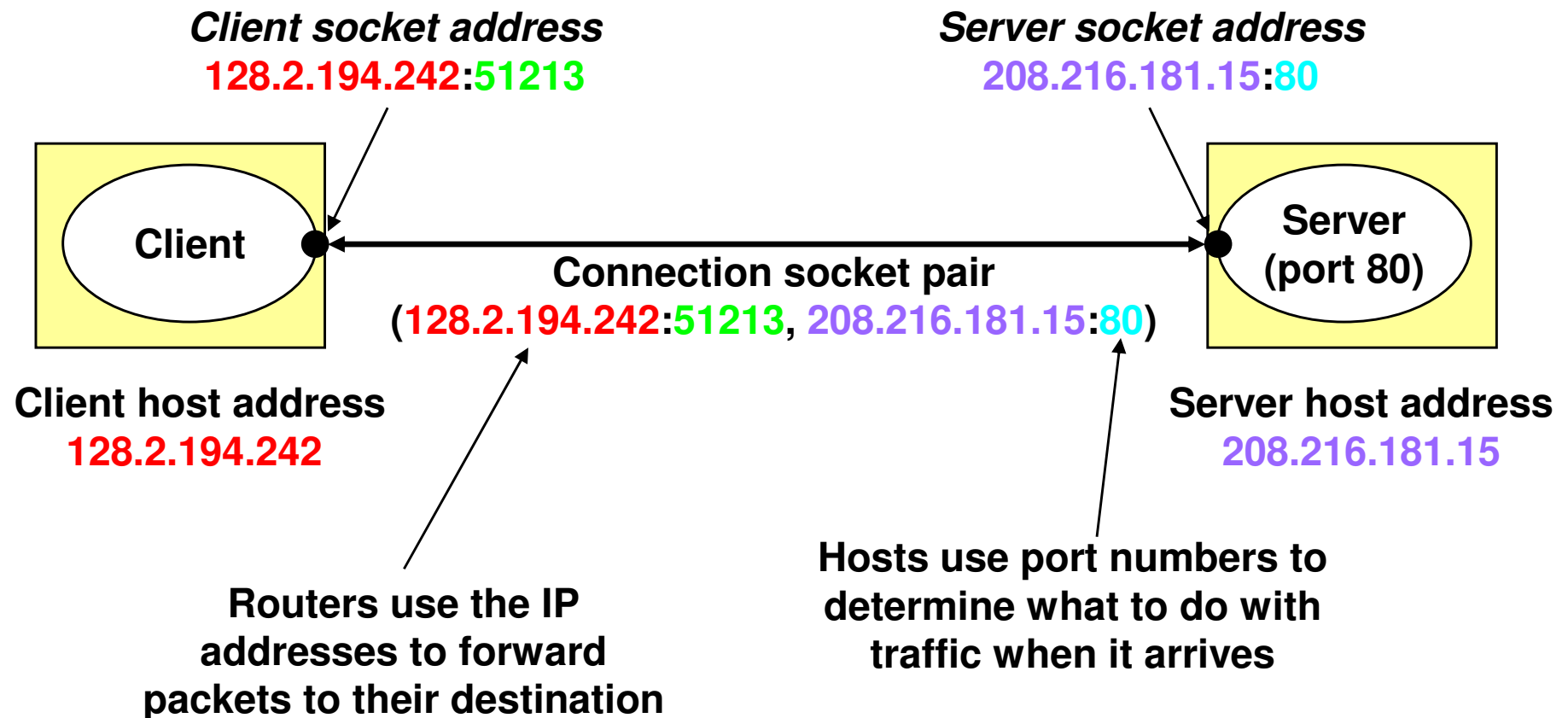
# Sending and Receiving Data

- Applications do not access the network directly
  - Use the socket abstraction
  - Send/receive data using operating system calls (i.e., write/read)
- What happens between the network interface and the application?

# Application-level Networking

- Applications use *sockets* to access the network
  - A socket is an endpoint of a network connection
  - Socket address is an IP address, port pair
- A port is a 16-bit integer identifying a process
  - Ephemeral port: Assigned automatically on host when host makes a connection request
  - Well-known port: Associated with some service provided by a host (e.g., port 80 is associated with Web servers)
- Socket addresses on two hosts uniquely identify communication
  - Socket pair
  - (cliaddr:cliport, servaddr:servport)

# Anatomy of an Internet Connection



# Sending Data using Write

## ■ In C:

```
ssize_t write(int fd, const void *buf, size_t count);
```

- File descriptor (fd) names a socket instead of a file
- Socket
  - Bound to a destination IP address
  - May also store additional information (TCP port number, etc.)

# Fragmentation

- Application can try to send any size buffer
- Operating system must fragment data into MTU-sized chunks!
- Packets are processed individually once they are fragmented
  - Processed in order (required for ordered protocols)
  - May be buffered to be dealt with later

# Finding the Destination

- The socket provides the destination IP address
- What else is needed?
  - Network interface data should be sent on (egress port?)
  - Information about next hop
    - IP address?
    - MAC address?
- Is this different than a router?

# Routing Table

```
rixner@nf-server1:~$ route
```

```
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
10.143.202.0	*	255.255.255.0	U	0	0	0	priv2
10.143.203.0	*	255.255.255.0	U	0	0	0	priv3
10.143.204.0	*	255.255.255.0	U	0	0	0	priv4
10.143.205.0	*	255.255.255.0	U	0	0	0	priv5
10.143.206.0	*	255.255.255.0	U	0	0	0	priv6
192.168.0.0	*	255.255.255.0	U	0	0	0	public0
default	comp519	0.0.0.0	UG	100	0	0	public0

```
rixner@nf-server1:~$
```

Destination

Next Hop

Subnet mask

Interface  
(output port)

*How does this differ from a router's routing table?*

# ARP Table

```
rixner@nf-server1:~$ arp
```

Address	HWtype	HWaddress	Flags	Mask	Iface
comp519	ether	00:E0:81:5A:71:AB	C		public0
10.143.206.194	ether	00:0D:BD:05:5A:00	C		priv6
10.143.206.2	ether	00:07:E9:A0:17:DB	C		priv6
10.143.206.130	ether	00:10:18:1C:5C:A1	C		priv6

```
rixner@nf-server1:~$
```

IP Address



Ethernet Address



*How does this differ from a router's ARP table?*

# Generating Packet Headers

- Operating system must generate the IP and Ethernet headers
- IP
  - Most header fields are obvious (destination IP address, length, protocol, etc.)
  - What about TTL, Identification, TOS, source IP address?
- Ethernet header
  - Destination: Next hop MAC address
  - Source: ?

# Checksum Calculation

- Cheap in hardware, expensive in software
  - Have to touch all of the packet data!
  - Nothing else in the OS needs to touch the packet data, so probably not in cache
- Let the NIC do it!
  - Modern network interfaces support checksum offloading
  - The operating system tells the NIC whether or not it wants it to compute the checksum



# NIC: Host Communication

- Direct Memory Access (DMA)
  - Transfer packets to/from host memory
  - Buffer locations provided by host device driver
- Interrupts
  - Notify host that packets have been sent
  - Notify host that packets have arrived



# NIC: Internal Processing

- Checksum offloading
  - Computes IP/TCP/UDP checksums
  - Speeds up network stack in operating system
- TCP segmentation offloading
  - Advertise large (64KB) MTU to operating system
  - Segment packets to real (1500B) MTU on NIC



# NIC: Network Access

## ■ PHY

- Transceiver for the physical medium
- Performs digital/analog conversion

## ■ MAC

- Transmits/receives data to/from the network
- Controls packet framing (e.g., Ethernet preamble, Ethernet CRC checksum, etc.)

# Receiving Data using Read

- In C:

```
ssize_t read(int fd, void *buf, size_t count);
```

- File descriptor (fd) names a socket instead of a file
- Socket
  - Bound to an IP address
  - May also store additional information (TCP port number, etc.)

# Data Reassembly

- The system may receive data that has been fragmented or reordered
  - Depending on the protocol, the operating system may “undo” this
  - UDP messages are reassembled
  - TCP data is forced to arrive in order
- Reliability (TCP)
  - Only ordered data is acknowledged to the sender and delivered to the application
  - Sender resends all unacknowledged data

# “Forwarding”

- Unlike a router, received data is not forwarded to another network
- Based on IP addresses, protocols, and ports, data is associated with a socket
  - Stored in a “socket buffer”
  - Delivered to an application when it *reads* from that socket



# Converting a Host into a Router

- Much of the functionality is there
- Simply take received data, and send it back out
  - The send path basically has all the necessary routing functionality
  - Routing table, ARP table, etc.
- The trick is making this *fast*