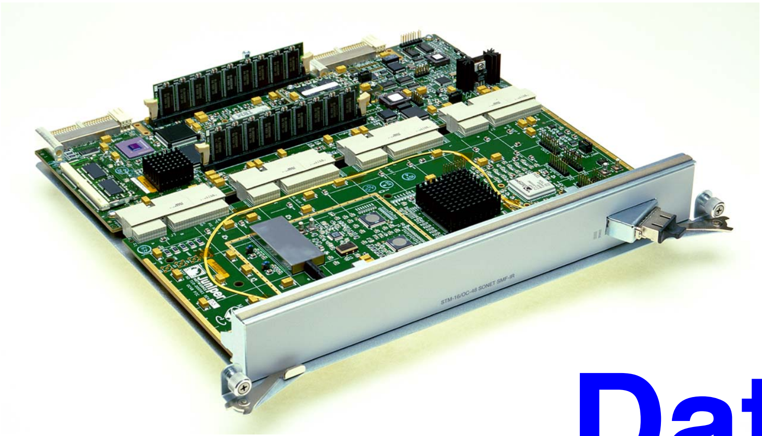
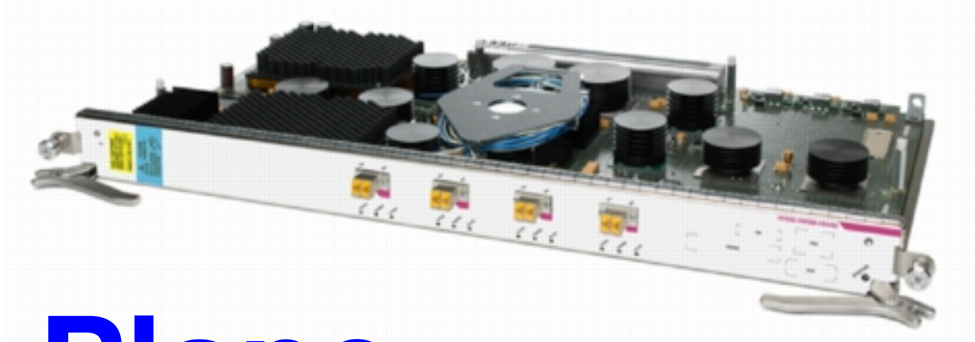




Data Plane

Jennifer Rexford

Fall 2010 (TTh 1:30-2:50 in COS 302)

COS 561: Advanced Computer Networks

http://www.cs.princeton.edu/courses/archive/fall10/cos561/

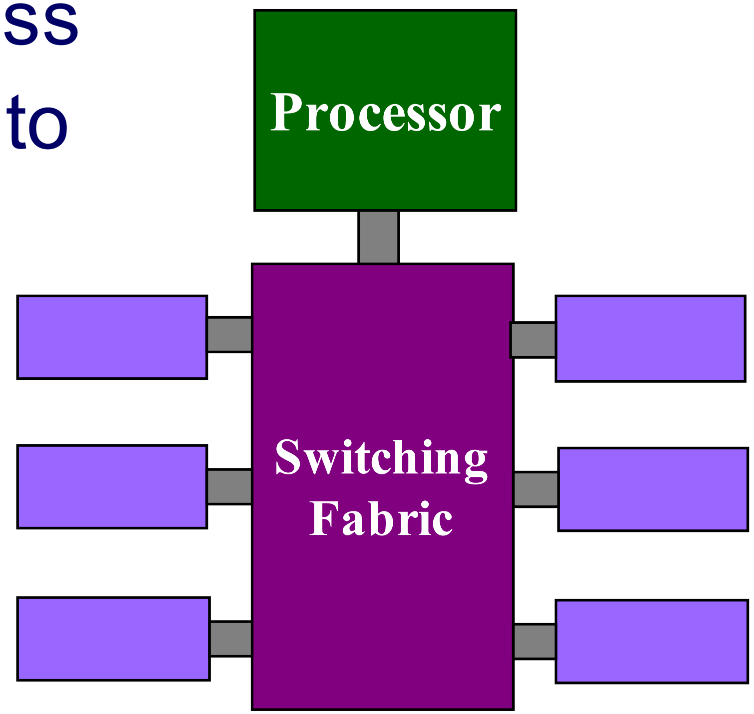
Data Plane

- Streaming algorithms that act on packets
  - Matching on some bits, taking a simple action
  - … at behest of control and management plane
- Wide range of functionality
  - Forwarding
  - Access control
  - Mapping header fields
  - Traffic monitoring
  - Buffering and marking
  - Shaping and scheduling
  - Deep packet inspection

Packet Forwarding

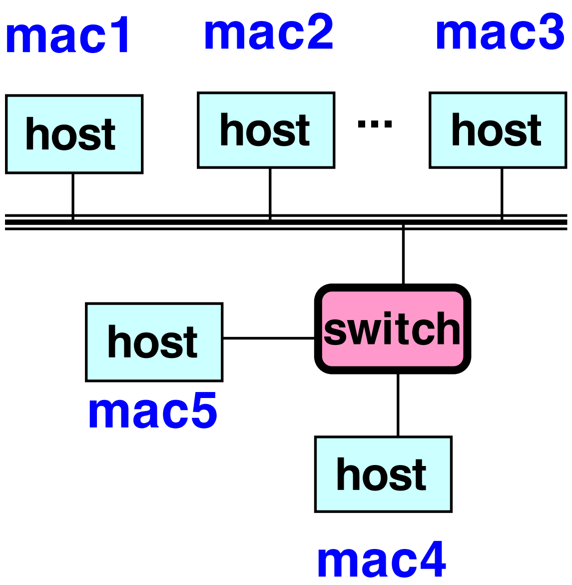
Packet Forwarding

- Control plane computes a forwarding table
  - Maps destination address(es) to an output link
- Handling an incoming packet
  - Match: destination address
  - Action: direct the packet to the chosen output link
- Switching fabric
  - Directs packet from input link to output link



Switch: Match on Destination MAC

- MAC addresses are location independent
  - Assigned by the vendor of the interface card
  - Cannot be aggregated across hosts in the LAN

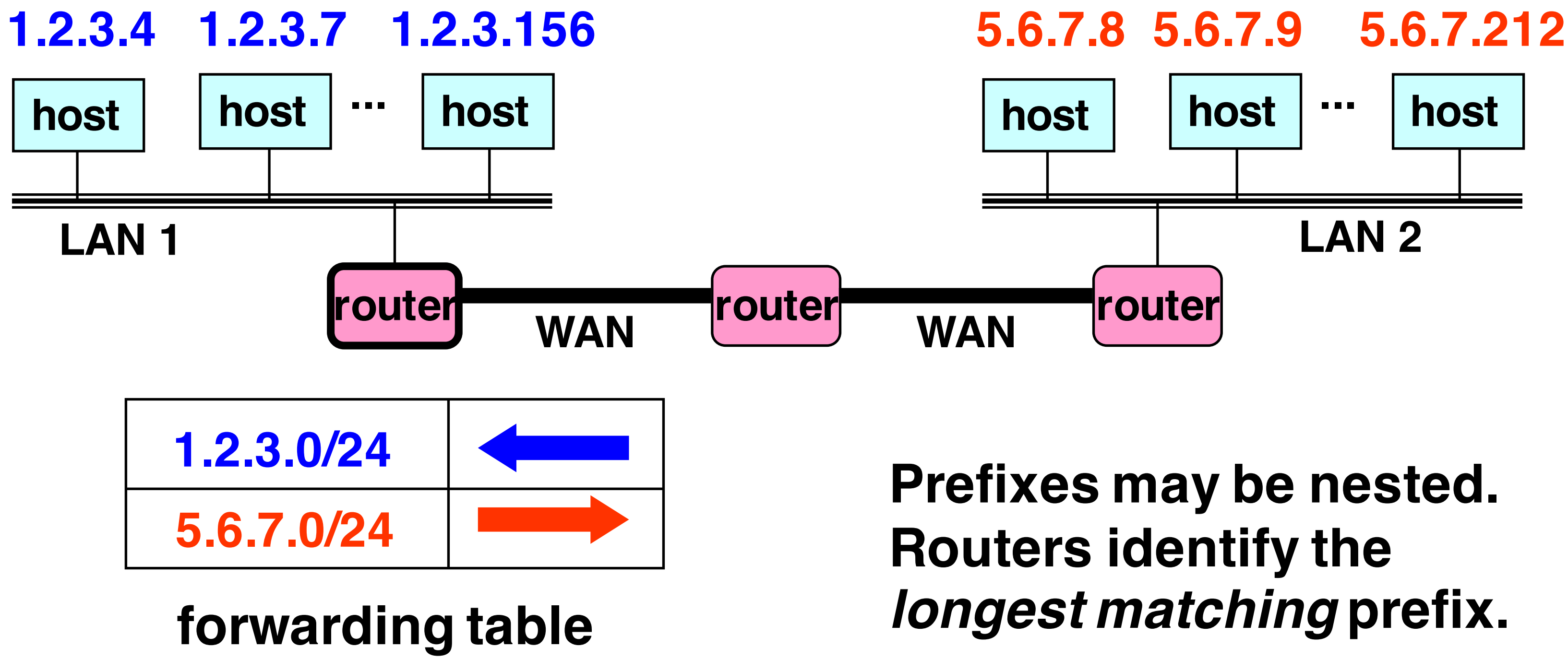


mac1	↑
mac2	↑
mac3	↑
mac4	←
mac5	↓

Implemented using a hash table or a content addressable memory.

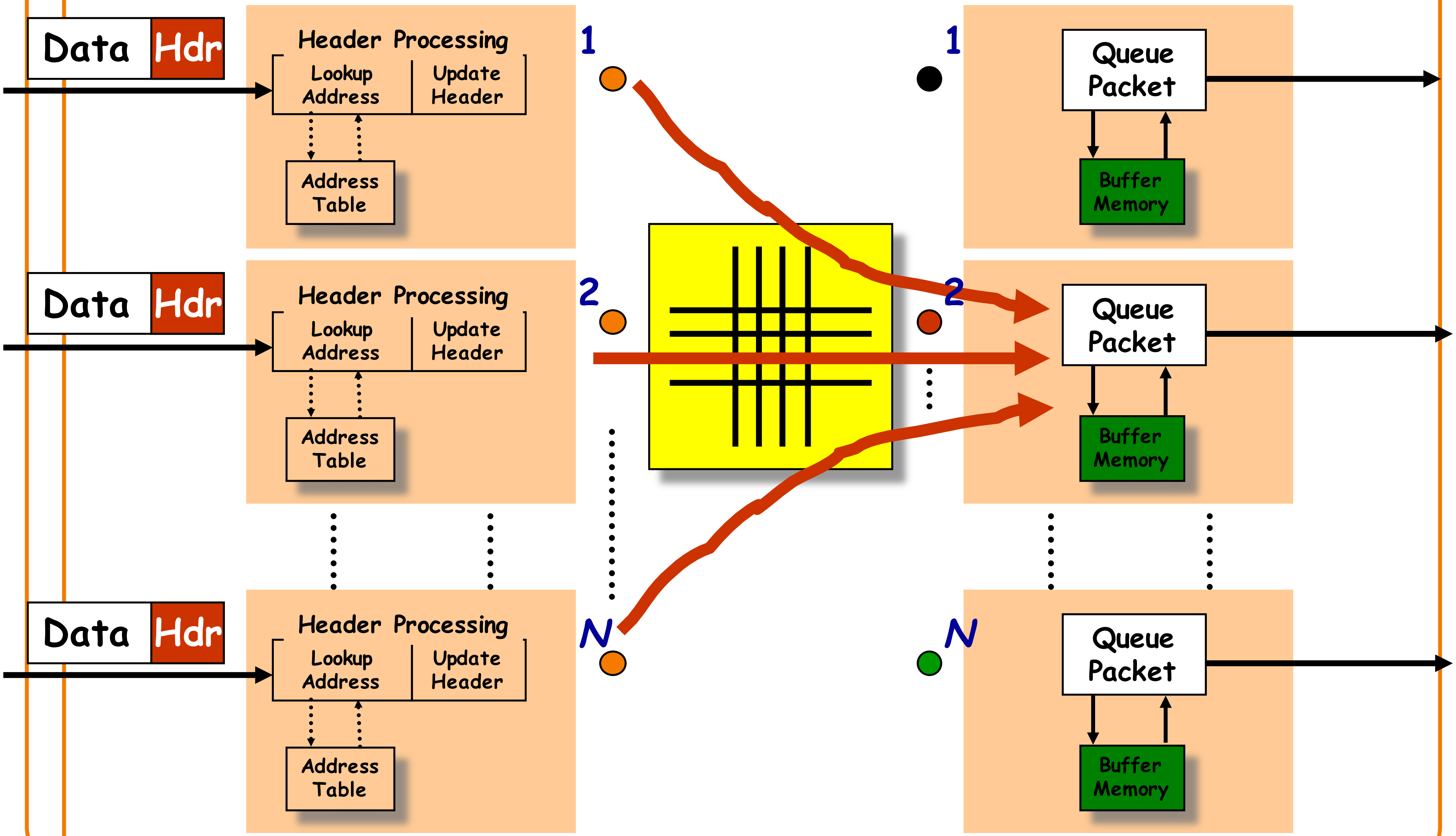
IP Routers: Match on IP Prefix

- IP addresses grouped into common subnets
  - Allocated by ICANN, regional registries, ISPs, and within individual organizations
  - Variable-length prefix identified by a mask length



Prefixes may be nested. Routers identify the *longest matching* prefix.

Switch Fabric: From Input to Output

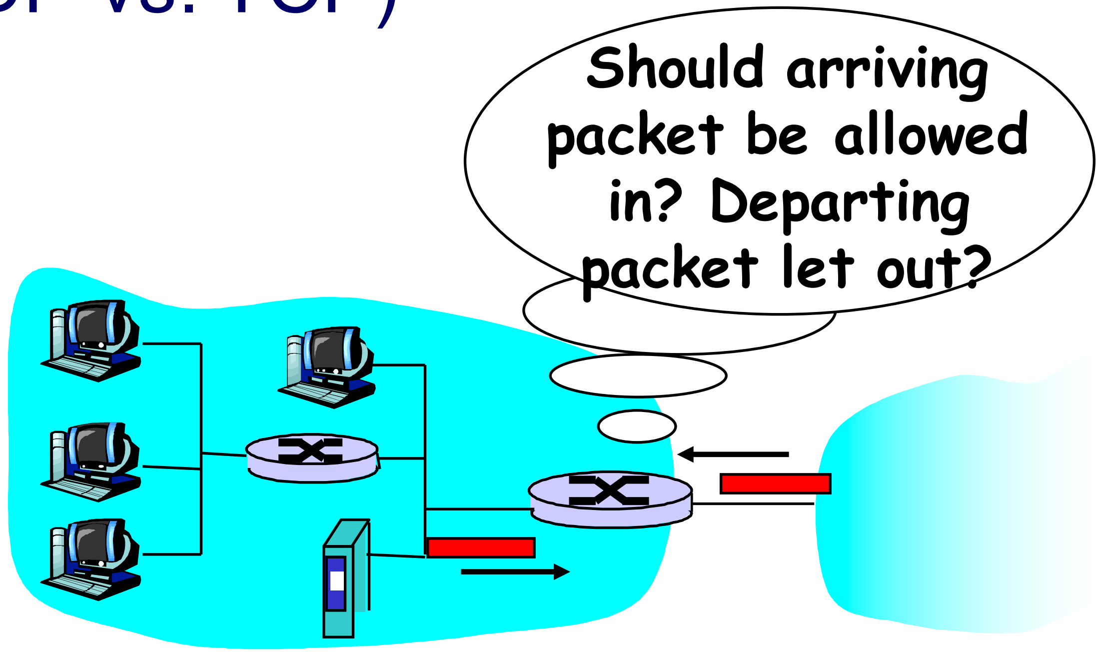


Access Control

Access Control: Packet Filtering

- "Five tuple" for access control lists (ACLs)
  - Source and destination IP addresses
  - TCP/UDP source and destination ports
  - Protocol (e.g., UDP vs. TCP)

- Can be more sophisticated
  - E.g., block all TCP SYN packets from outside hosts



Applying Access Control Lists

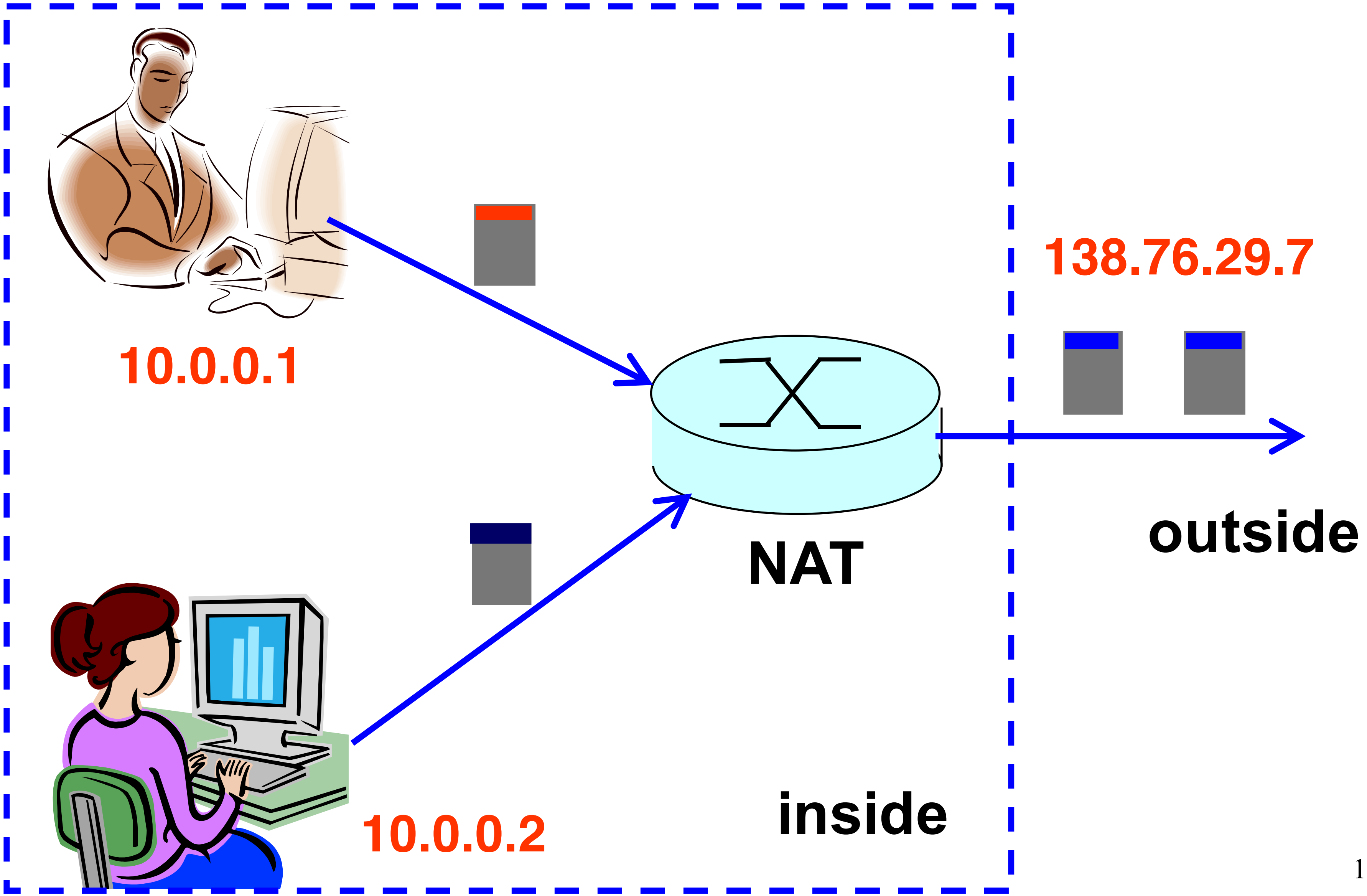
- Ordered list of “accept/deny” clauses
  - A clause can have wild cards
  - Clauses can overlap
  - … so order matters
- Packet classification
  - Given all of the fields
  - … identify the match with the highest priority

Src=1.2.3.4, Dest=5.6.7.8	Deny
Dest=1.2.3.*	Allow
Dest=1.2.3.8, Dport!=53	Deny
Src=1.2.3.7, Dport=100	Allow
Dport=100	Deny

- Two approaches
  - Clever algorithms for multi-dimensional classification
  - Ternary Content Addressable Memories (TCAMs)

Mapping Header Fields

Network Address Translation (NAT)

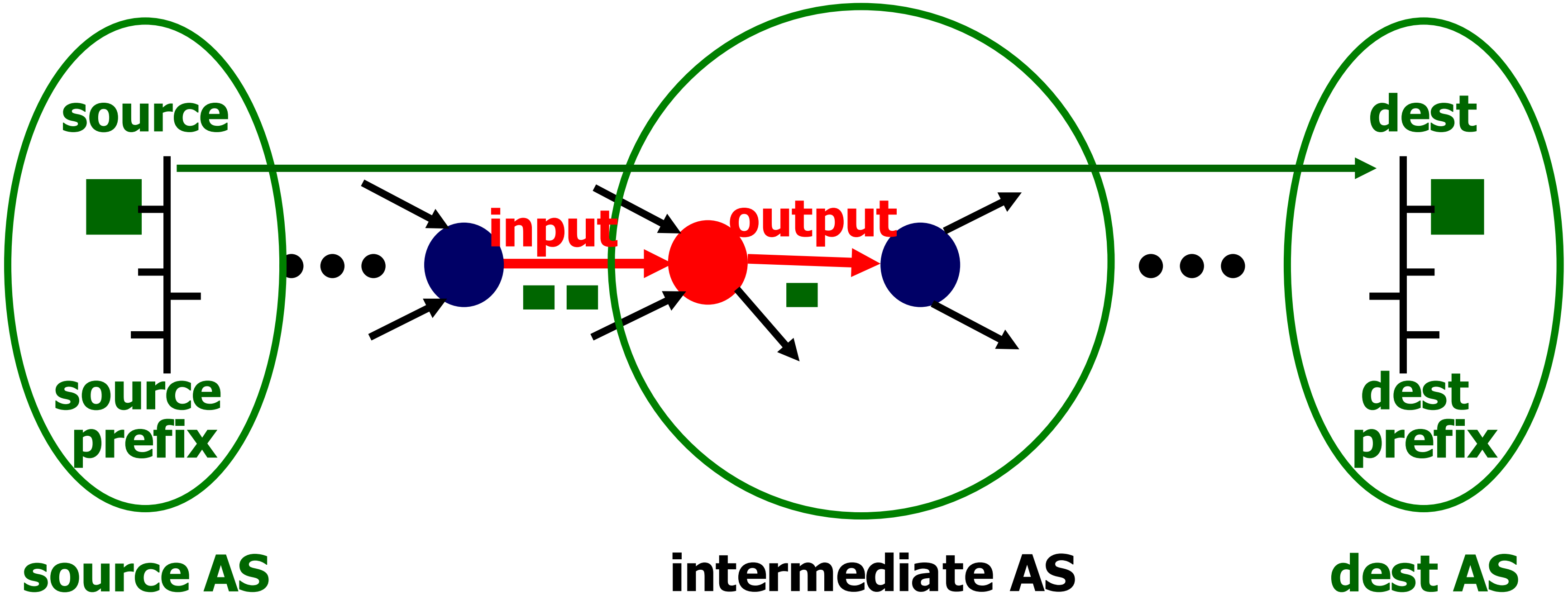


Mapping Addresses and Ports

- Remap IP addresses and TCP/UDP port numbers
  - Addresses: between end-host and NAT addresses
  - Port numbers: to ensure each connection is unique
- Create table entries as packets arrive
  - Src 10.0.0.1, Sport 1024, Dest 1.2.3.4, Dport 80
    - Map to Src 138.76.29.7, Sport 1024, Dest 1.2.3.4, Dport 80
  - Src 10.0.0.2, Sport 1024, Dest 1.2.3.4, Dport 80
    - Map to Src 138.76.29.7, Sport 1025, Dest 1.2.3.4, Dport 80
- Challenges
  - When to remove the entries
  - Running services behind a NAT
  - What if both ends of a connection are behind NATs

Traffic Monitoring

Observing Traffic Passing Through



- Applications of traffic measurement
  - –Usage-based billing
  - –Network engineering
  - –Detecting anomalous traffic

Passive Traffic Monitoring

- Counting the traffic
  - Match based on fields in the packet header
  - … and update a counter of #bytes and #packets
- Examples
  - Link
  - IP prefixes
  - TCP/UDP ports
  - Individual “flows”

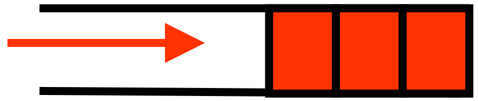
Dest Prefix	#Packets	#Bytes
1.2.3.0/24	3	1500
7.8.0.0/16	10	13000
8.0.0.0/8	100	85020
7.7.6.0/23	1	40

- Challenges
  - Identify traffic aggregates in advance vs. reactively
  - Summarizing other information (e.g., time, TCP flags)
  - Not knowing if you see all packets in a connection

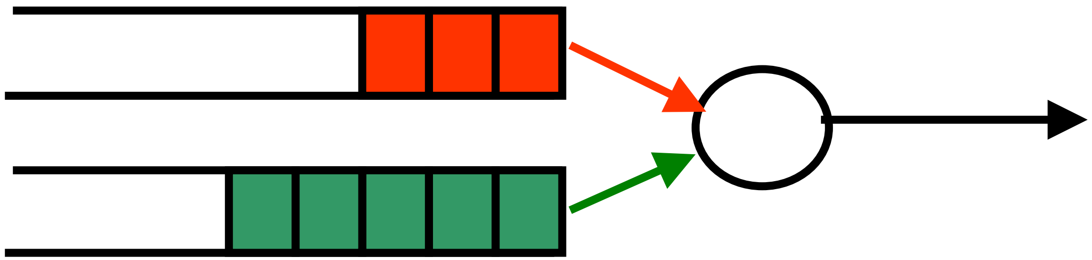
Resource Allocation: Buffering, Scheduling, Shaping, and Marking

Buffering

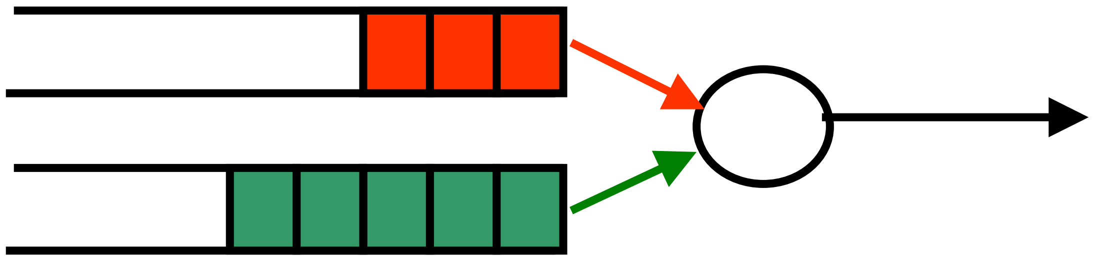
- Drop-tail FIFO queue
  - Packets served in the order they arrive
  - … and dropped if queue is full



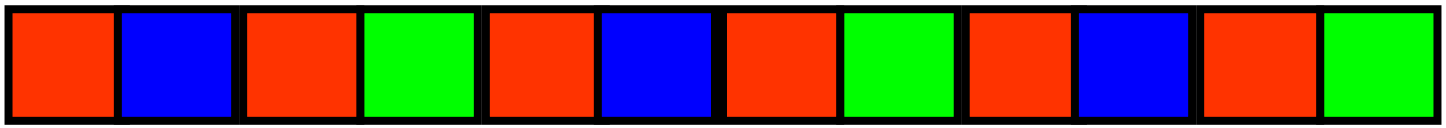
- Random Early Detection (RED)
  - When the buffer is nearly full
  - … drop or mark some packets to signal congestion
- Multiple classes of traffic
  - Separate FIFO queue for each flow or traffic class
  - … with a link scheduler to arbitrate between them



Link Scheduling



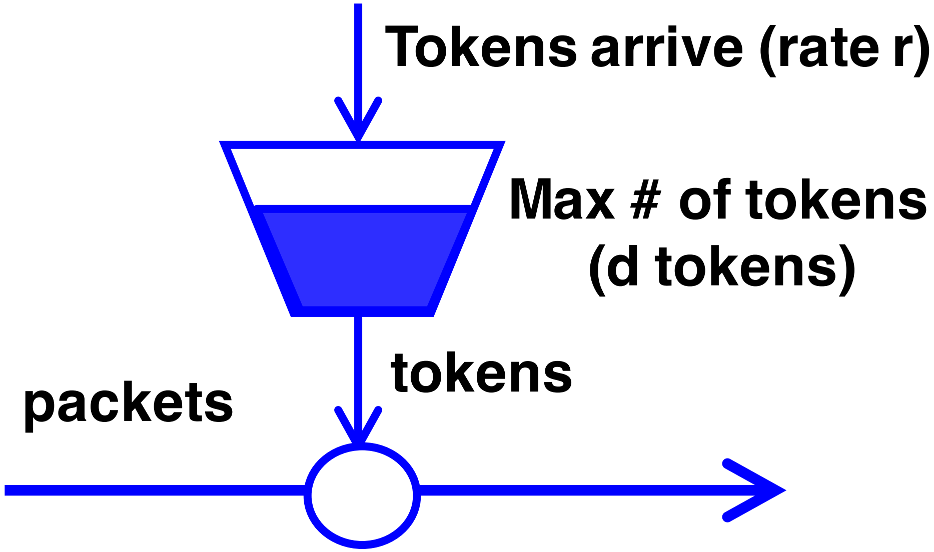
- Strict priority
  - Assign an explicit rank to the queues
  - … and serve the highest-priority backlogged queue
- Weighted fair scheduling
  - Interleave packets from different queues
  - …in proportion to weights



50% red, 25% blue, 25% green

Traffic Shaping

- Force traffic to conform with a profile
  - To avoid congesting downstream resources
  - To enforce a contract with the customer
- Leaky-bucket shaping
  - Can send at rate r and intermittently burst
  - Parameters: token rate r and bucket depth d



A leaky-bucket shaper for each flow or traffic class

Traffic Classification and Marking

- Mark a packet to influence handling downstream
  - Early Congestion Notification (ECN) flag
  - Type-of-Service (ToS) bits
- Ways to set the ToS bits
  - End host sets the bits based on the application
    - But, then the network must trust (or bill!) the end host
  - Network sets the bits based on traffic classes
    - But, then the network needs to know how to classify packets
- Identifying traffic classes
  - Packet classification based on the "five tuple"
  - Rate limits, with separate mark for "out of profile" traffic

Generalizing the Data Plane

Many Boxes, But Similar Functions

- Router
  - Forward on destination IP address
  - Access control on the "five tuple"
  - Link scheduling and marking
  - Monitoring traffic
  - Deep packet inspection
- Switch
  - Forward on destination MAC address
- Firewall
  - Access control on "five tuple" (and more)
- NAT
  - Mapping addresses and port numbers
- Shaper
  - Classify packets
  - Shape or schedule
- Packet sniffer
  - Monitoring traffic

OpenFlow

- Match
  - Match on a subset of bits in the packet header
  - E.g., key header fields (addresses, port numbers, etc.)
  - Well-suited to capitalize on TCAM hardware
- Action
  - Perform a simple action on the matching packet
  - E.g., forward, flood, drop, rewrite, count, etc.
- Controller
  - Software that installs rules and reads counts
  - ... and handles packets the switch cannot handle

Programmable Data Plane

- Programmable data plane
  - Arbitrary customized packet-handling functionality
  - Building a new data plane, or extending existing one
- Speed is important
  - Data plane in hardware or in the kernel
  - Streaming algorithms the handle packets as they arrive
- Two open platforms
  - Click: software data plane in user space or the kernel
  - NetFPGA: hardware data plane based on FPGAs
- Lots of ongoing research activity…

Discussion

- Past experiences using Click?
- Trade-offs between customizability and performance?
- What data-plane model would be “just enough” for most needs, but still fast and inexpensive?
- Ways the Internet architecture makes data-plane functionality more challenging?
- Middleboxes: an abomination or a necessity?

Click Modular Router
(backup slides)

Click Motivation

- Flexibility
  - Add new features and enable experimentation
- Openness
  - Allow users/researchers to build and extend
  - (In contrast to most commercial routers)
- Modularity
  - Simplify the composition of existing features
  - Simplify the addition of new features
- Speed/efficiency
  - Operation (optionally) in the operating system
  - Without the user needing to grapple with OS internals

Router as a Graph of Elements

- Large number of small elements
  - Each performing a simple packet function
  - E.g., IP look-up, TTL decrement, buffering
- Connected together in a graph
  - Elements inputs/outputs snapped together
  - Beyond elements in series to a graph
  - E.g., packet duplication or classification
- Packet flow as main organizational primitive
  - Consistent with data-plane operations on a router
  - (Larger elements needed for, say, control planes)

Click Elements: Push vs. Pull

- Packet hand-off between elements
  - Directly inspired by properties of routers
  - Annotations on packets to carry temporary state
- Push processing
  - Initiated by the source end
  - E.g., when an unsolicited packet arrives (e.g., from a device)
- Pull processing
  - Initiated by the destination end
  - E.g., to control timing of packet processing (e.g., based on a timer or packet scheduler)

Click Language

- Declarations
  - Create elements
- Connections
  - Connect elements

```
src :: FromDevice(eth0);
ctr :: Counter;
sink :: Discard;

src -> ctr;
ctr -> sink;
```

- Compound elements
  - Combine multiple smaller elements, and treat as single, new element to use as a primitive class
- Language extensions through element classes
  - Configuration strings for individual elements
  - Rather than syntactic extensions to the language

Handlers and Control Socket

- Access points for user interaction
  - Appear like files in a file system
  - Can have both read and write handlers
- Examples
  - Installing/removing forwarding-table entries
  - Reporting measurement statistics
  - Changing a maximum queue length
- Control socket
  - Allows other programs to call read/write handlers
  - Command sent as single line of text to the server
  - http://read.cs.ucla.edu/click/elements/controlsocket?s=llrpc

Example: EtherSwitch Element

- Ethernet switch
  - Expects and produces Ethernet frames
  - Each input/output pair of ports is a LAN
  - Learning and forwarding switch among these LANs
- Element properties
  - Ports: any # of inputs, and same # of outputs
  - Processing: push
- Element handlers
  - Table (read-only): returns port association table
  - Timeout (read/write): returns/sets TIMEOUT

http://read.cs.ucla.edu/click/elements/etherswitch

An Observation…

- Click is widely used
  – And the paper on Click is widely cited
- Click elements are created by others
  – Enabling an ecosystem of innovation
- Take-away lesson
  – Creating useful systems that others can use and extend has big impact in the research community
  – And brings tremendous professional value
  – Compensating amply for the time and energy ☺