

# Lecture 23:

## Scheduling and QoS

---

CSE 123: Computer Networks  
Stefan Savage





# Lecture 23 Overview

---

- Scheduling
  - ◆ (Weighted) Fair Queuing
- Quality of Service basics
  - ◆ Integrated Services
  - ◆ Differentiated Services

# Scheduling

---

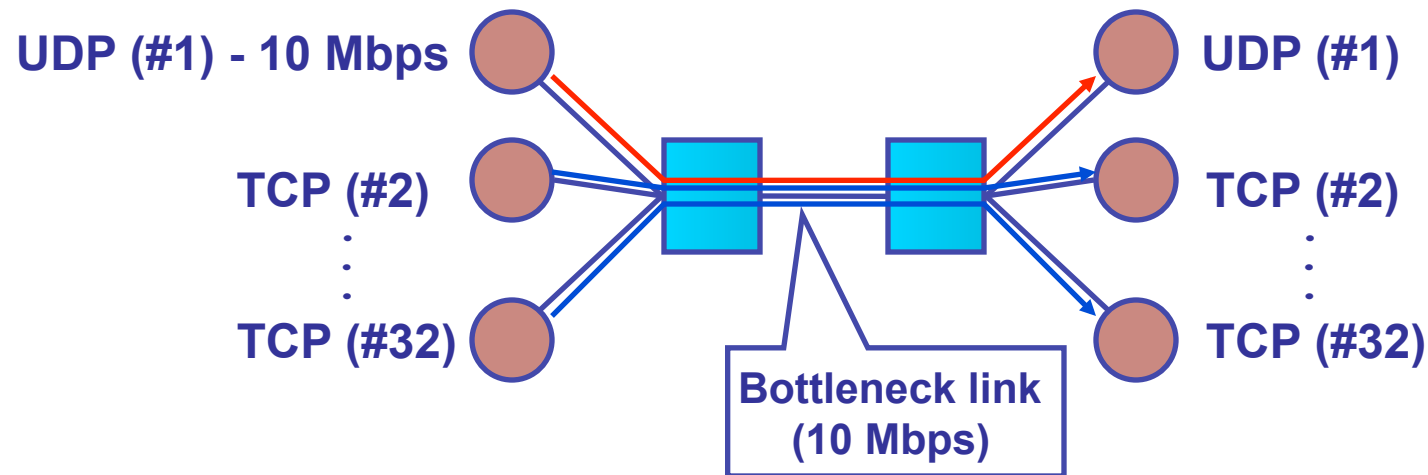


- So far we've done flow-based **traffic policing**
  - ◆ Limit the rate of one flow regardless of the load in the network
- In general, need **scheduling**
  - ◆ Dynamically allocate resources when multiple flows compete
  - ◆ Give each “flow” (or traffic class) own queue (at least theoretically)
- Weighted fair queuing
  - ◆ Proportional share scheduling
  - ◆ Schedule round-robins among queues in proportion to some weight parameter

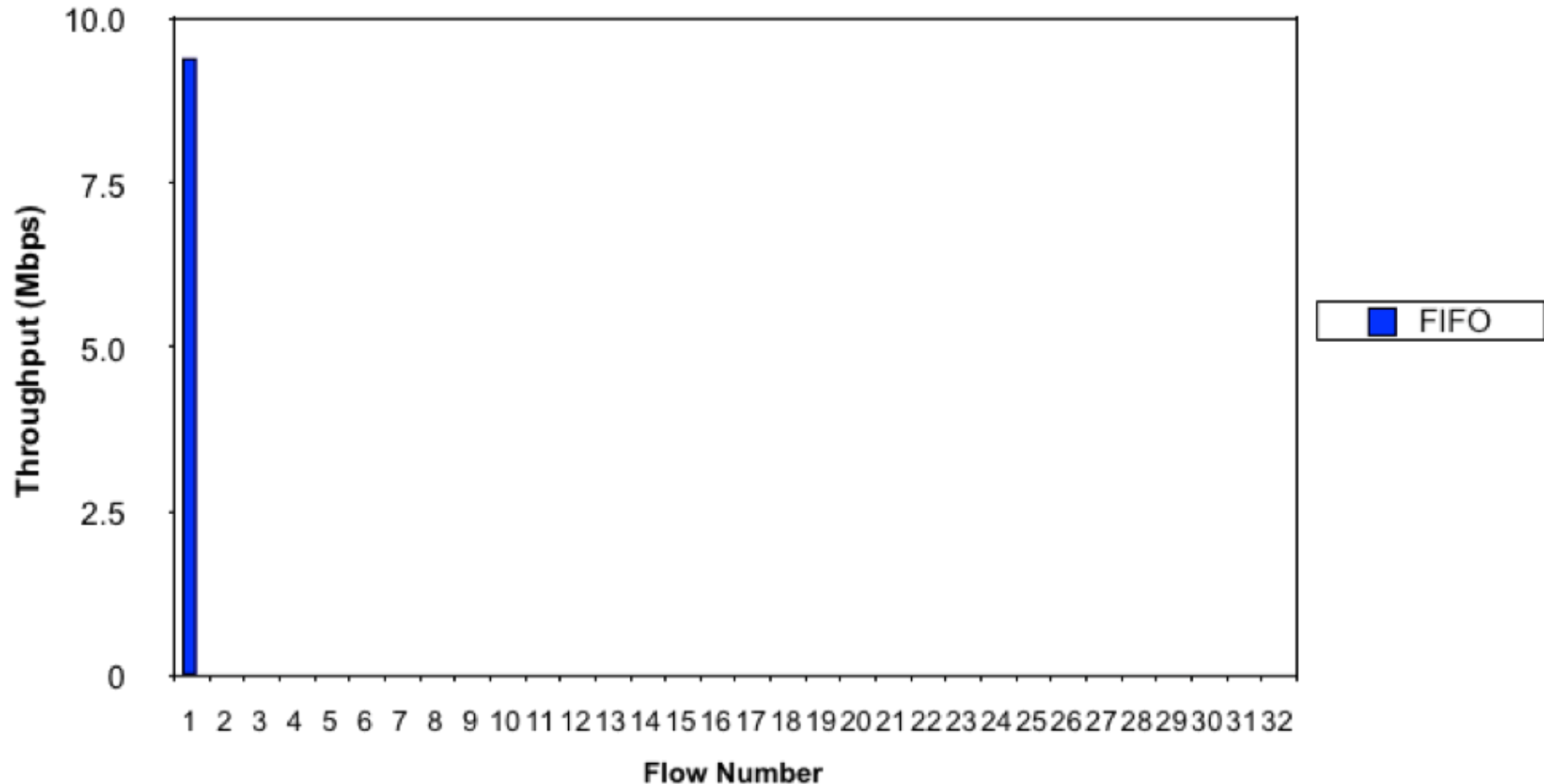


# Our Previous Example

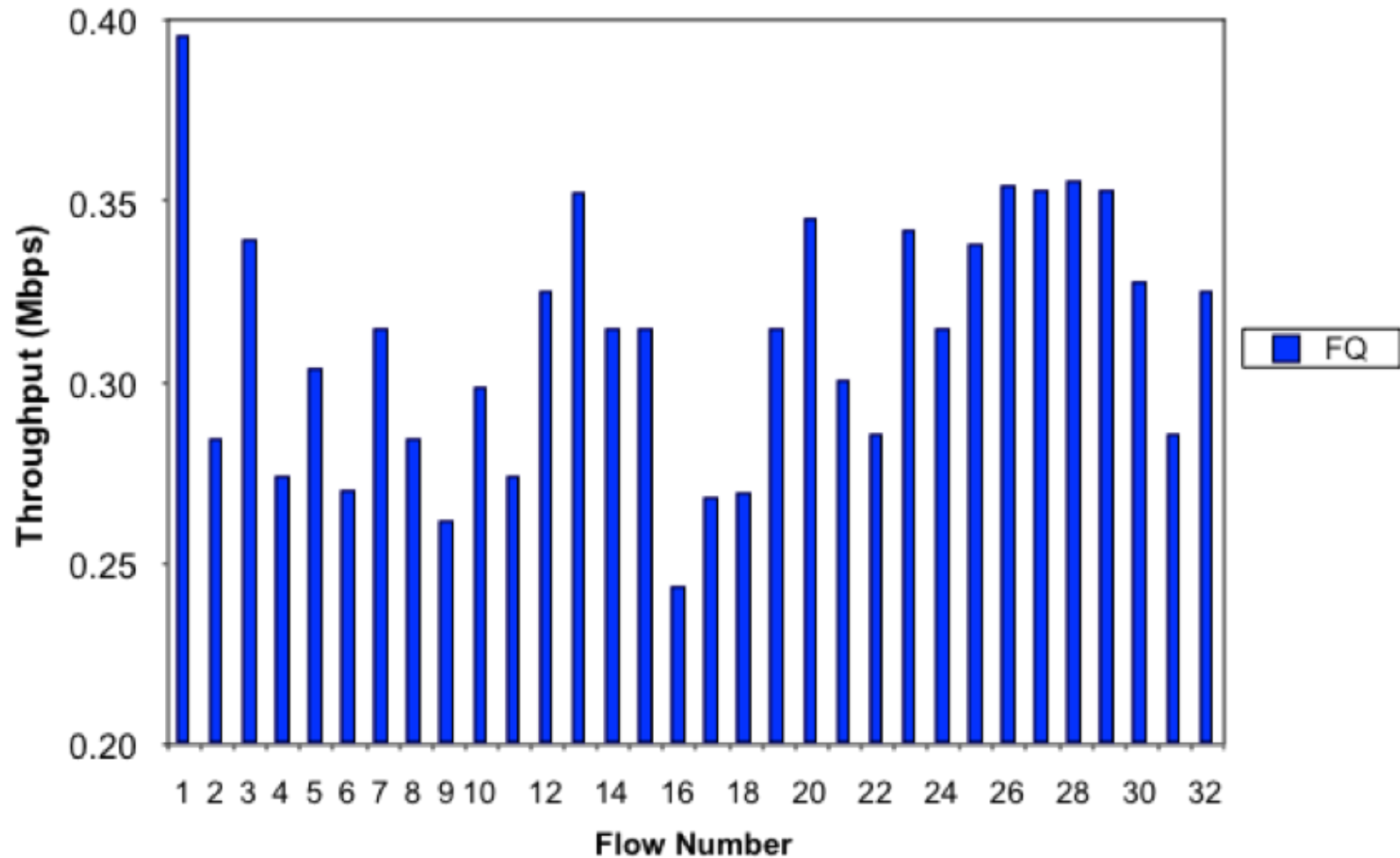
1 UDP (10 Mbps) and 31 TCPs sharing a 10 Mbps line



# UDP vs. TCP w/FIFO

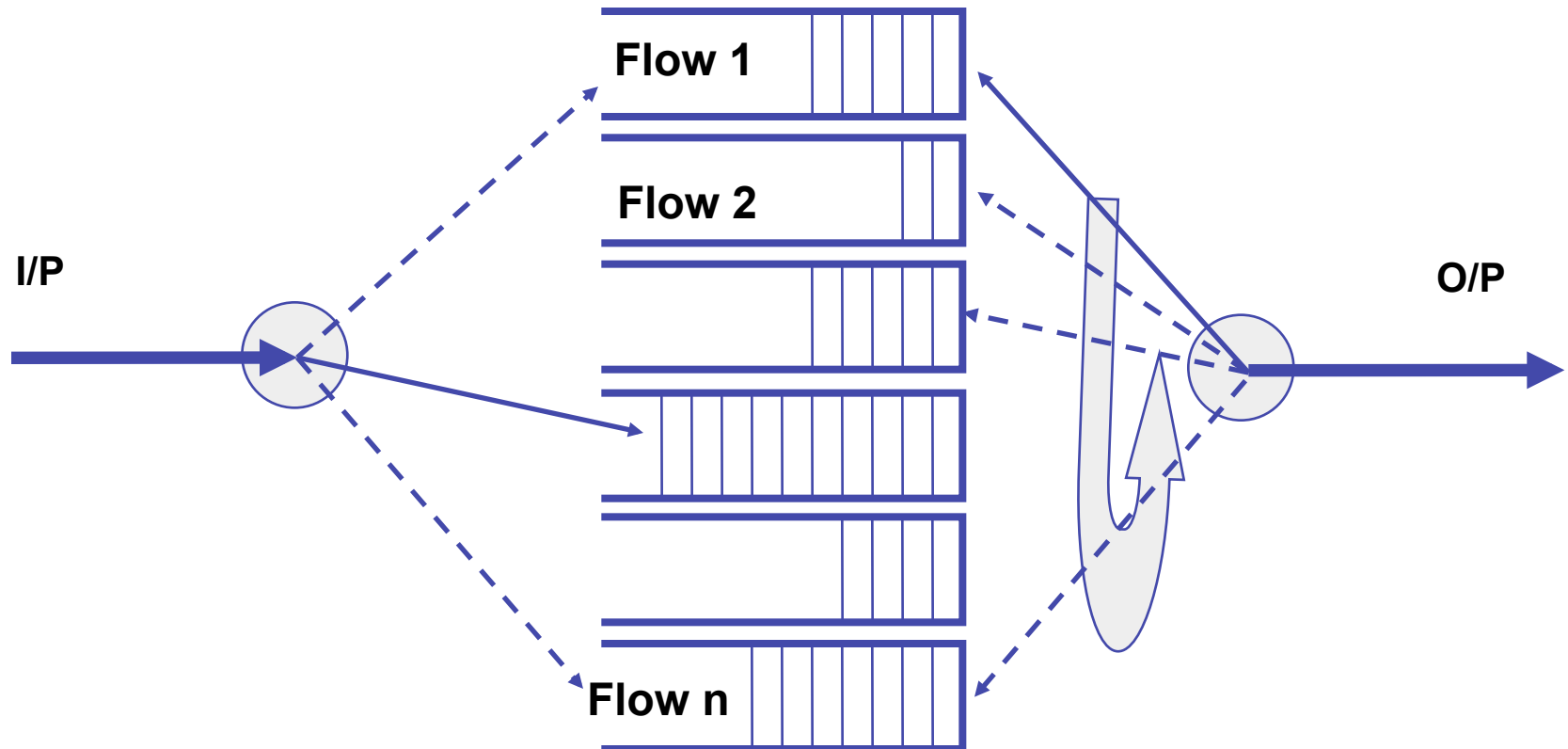


# TCP vs. UDP w/Fair Queuing





# (Weighted) Fair Queuing





# Fair Queuing

---

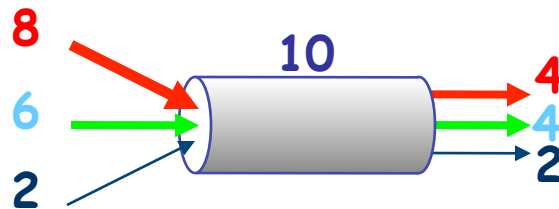
- Maintain a queue for each flow
  - ◆ What is a flow?
- Implements **max-min fairness**: each flow receives  $\min(r_i, f)$ , where
  - ◆  $r_i$  – flow arrival rate
  - ◆  $f$  – link fair rate (see next slide)
- **Weighted Fair Queuing** (WFQ) – associate a weight with each flow to divvy bandwidth up non-equally



# Fair Rate Computation

- If link congested, compute  $f$  such that

$$\sum_i \min(r_i, f) = C$$



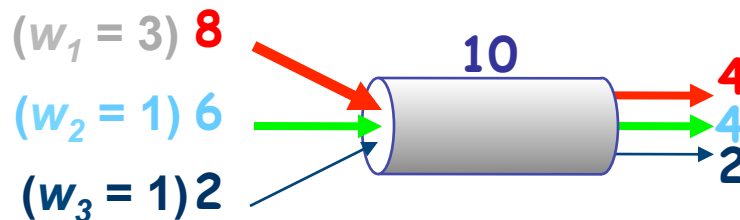
$f = 4$ :  
 $\min(8, 4) = 4$   
 $\min(6, 4) = 4$   
 $\min(2, 4) = 2$



# Another Example

- Associate a weight  $w_i$  with each flow  $i$
- If link congested, compute  $f$  such that

$$\sum_i \min(r_i, f \times w_i) = C$$



$$\begin{aligned} f &= 2: \\ \min(8, 2 \times 3) &= 6 \\ \min(6, 2 \times 1) &= 2 \\ \min(2, 2 \times 1) &= 2 \end{aligned}$$

Flow  $i$  is guaranteed to be allocated a rate  $\geq w_i \cdot C / (\sum_k w_k)$



If  $\sum_k w_k \leq C$ , flow  $i$  is guaranteed to be allocated a rate  $\geq w_i$



# Fluid Flow

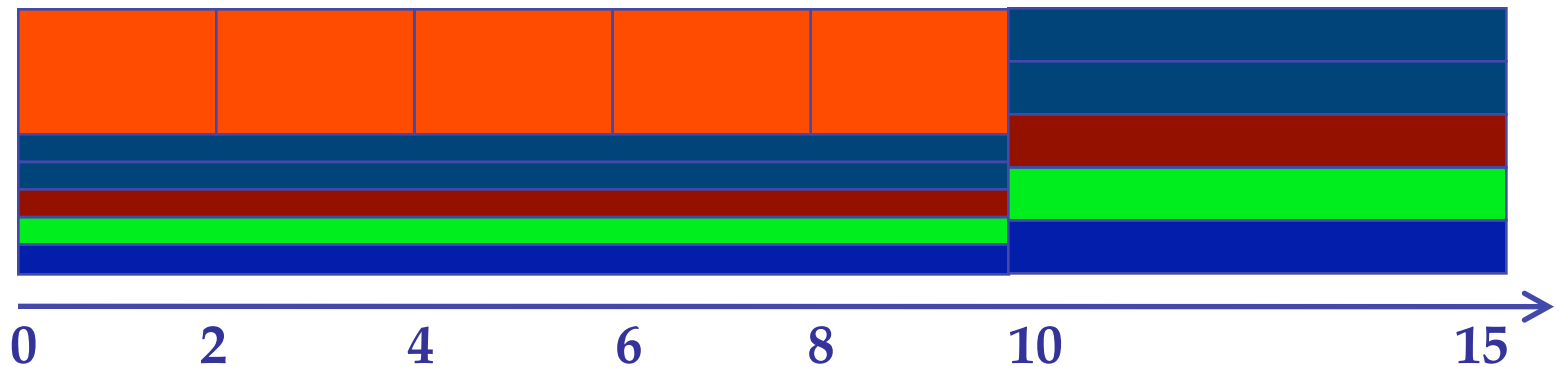
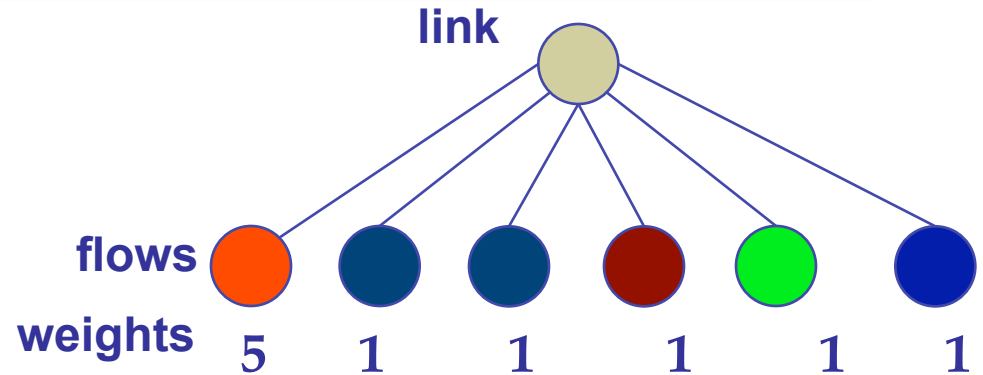
---

- Flows can be served one bit at a time
- WFQ can be implemented using bit-by-bit weighted round robin
  - ◆ During each round from each flow that has data to send, send a number of bits equal to the flow's weight



# Fluid Flow Example

- **Orange flow** has packets backlogged between time 0 and 10
- Other flows have packets continuously backlogged
- All packets have the same size



# Packet-Based Implementation

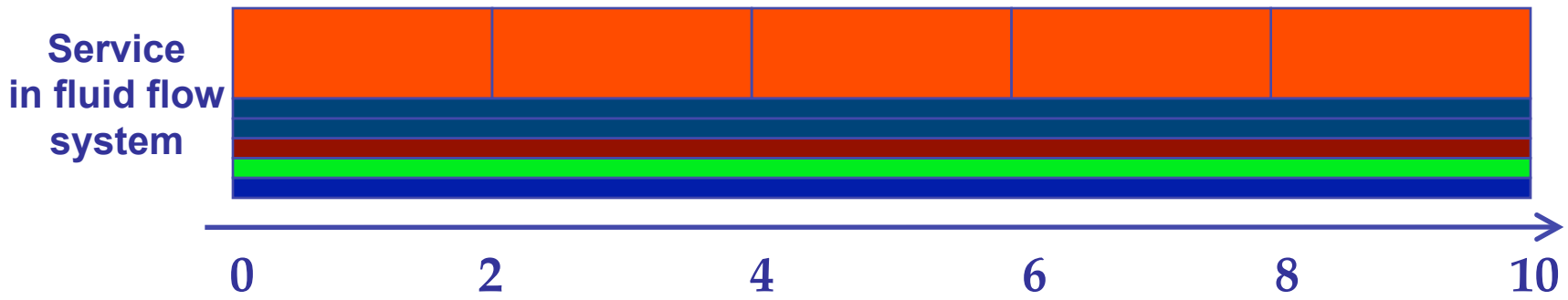
---



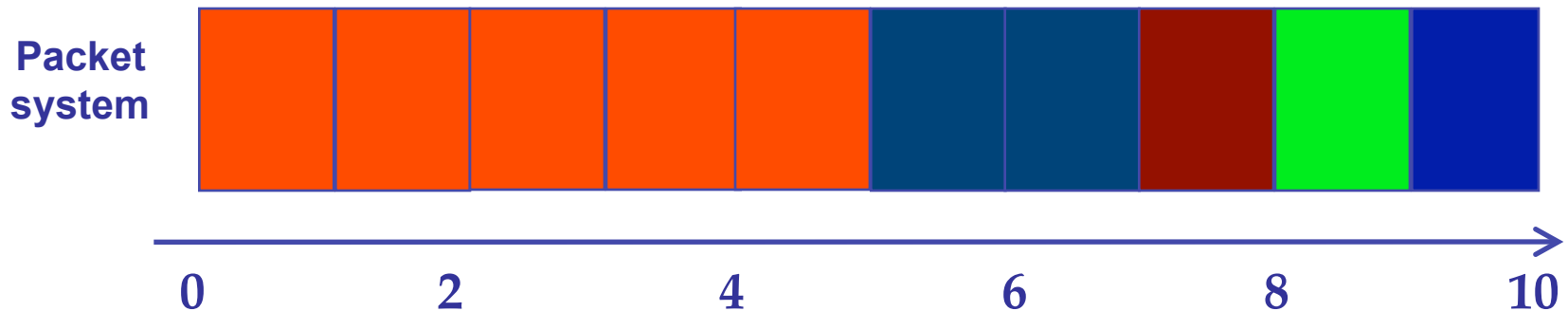
- Packet (Real) system: packet transmission cannot be preempted. Why?
- Solution: serve packets in the order in which they would have finished being transmitted in the fluid flow system



# Packet-Based Example



- Select the first packet that finishes in the fluid flow system





# Quality of Service (QoS)

---

- So far, we have assumed all traffic is equal and provided best effort delivery
  - ◆ Perhaps with enforcement to throttle non-responsive senders
- Not always best model. Why?
  - ◆ Application demands
    - » I want low-delay low-loss for phone service
    - » For backup, I just need bandwidth... don't care about delay
  - ◆ Market differentiation
    - » I want to sell better service for more money
  - ◆ Bandwidth management
    - » Don't let BitTorrent eat up all UCSD bandwidth

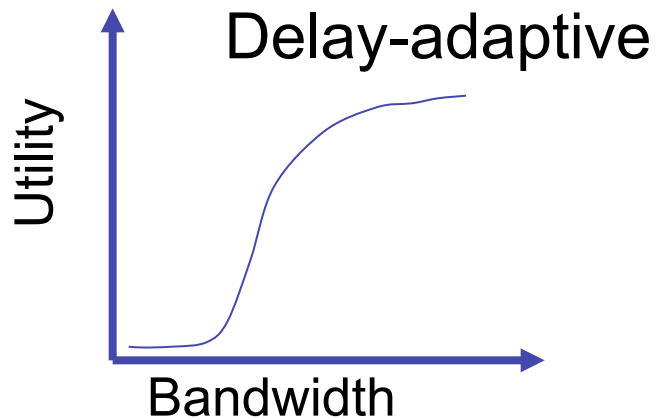
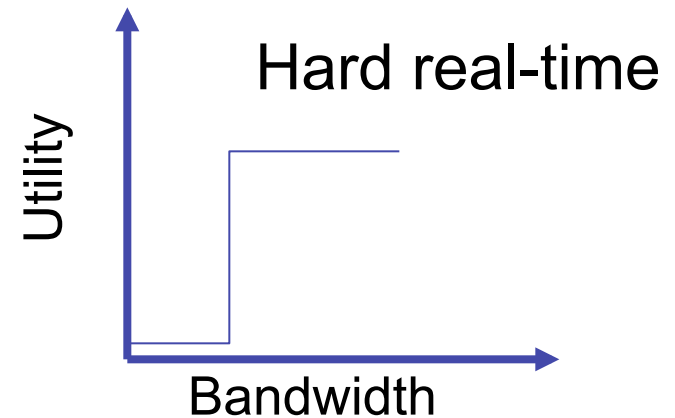
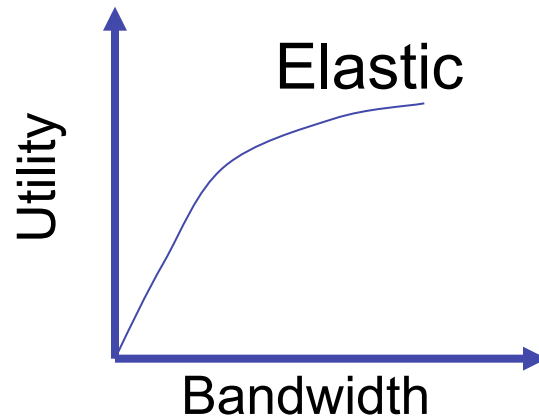


# Multimedia Applications

---

- Basic idea
  - ◆ Sample signal, packetize, transmit
  - ◆ Repeat in reverse at receiver
- Network Requirements (@ given load)
  - ◆ Delay
  - ◆ Jitter (variation in delay)
  - ◆ Packet loss
  - ◆ Exact parameters a function of interactivity demands, buffer capacity, retransmission time and loss tolerance
  - ◆ However... as a rule they want **more**

# Different Demands





# Packet Classification

---

- Want to treat some traffic better/worse than others
  - ◆ How to identify the more important traffic?
  - ◆ How much better do we want to treat it?
  - ◆ How do we actually treat it better?
- Router **classifies** based on packet header
  - ◆ Aggregates
    - » From particular network (IP src address)
    - » For particular protocol (e.g., port 80 traffic)
  - ◆ Individual network flows
    - » 5-tuple (src, dst, src port, dst port, protocol)
  - ◆ Special header field that indicates traffic “type”



# Service Classes

---

- Best-effort
  - ◆ Vanilla IP
- Differentiated service
  - ◆ Bronze, Silver, Gold, etc... (effectively priorities, **up to** some amount of bandwidth per time)
  - ◆ E.g., best service up to 10Mbps, then best effort
- Predicted service (soft real-time)
  - ◆ Network guarantees good performance on average
  - ◆ Application *promises* only send as fast as negotiated
- Guaranteed service (hard real-time)
  - ◆ Network guarantees good performance always
  - ◆ Application promises only send as fast as negotiated



# How to Specify?

---

- Kind of service (service class)
- Specify “flowspec” for data flow limits
  - ◆ Tspec: describes the flow’s traffic characteristics
    - » Average bandwidth + burstiness (contract with ISP)
  - ◆ Rspec: describes the service requested from the network (e.g., delay target)
- Interface can be interactive (ask network) or via business interface (ask salesman)
  - ◆ Can say no
  - ◆ If yes, then use scheduling mechanisms in routers (not FIFO anymore) to deliver





# Network-wide QoS

---

- Integrated services
  - Motivated by need for end-to-end guarantees
  - On-line negotiation of per-flow requirements
  - End-to-end per-router negotiation of resources
  - Complex
- Differentiated services
  - Motivated by economics (multi-tier pricing)
  - No per-flow state
  - Not end-to-end and not guaranteed services
  - Simple

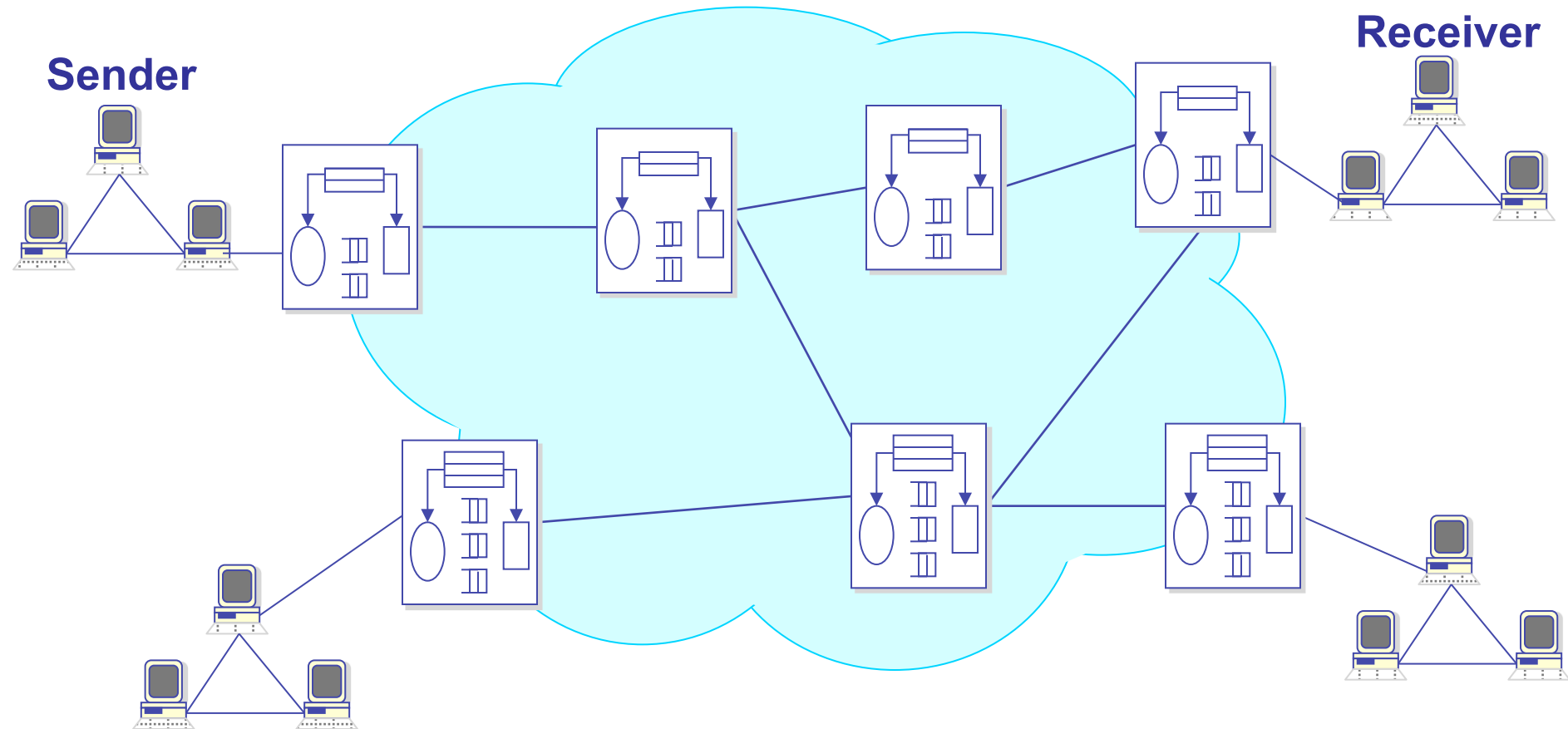


# Integrated Services

- Example: guarantee 1MBps and  $< 100$  ms delay to a flow

**Sender**

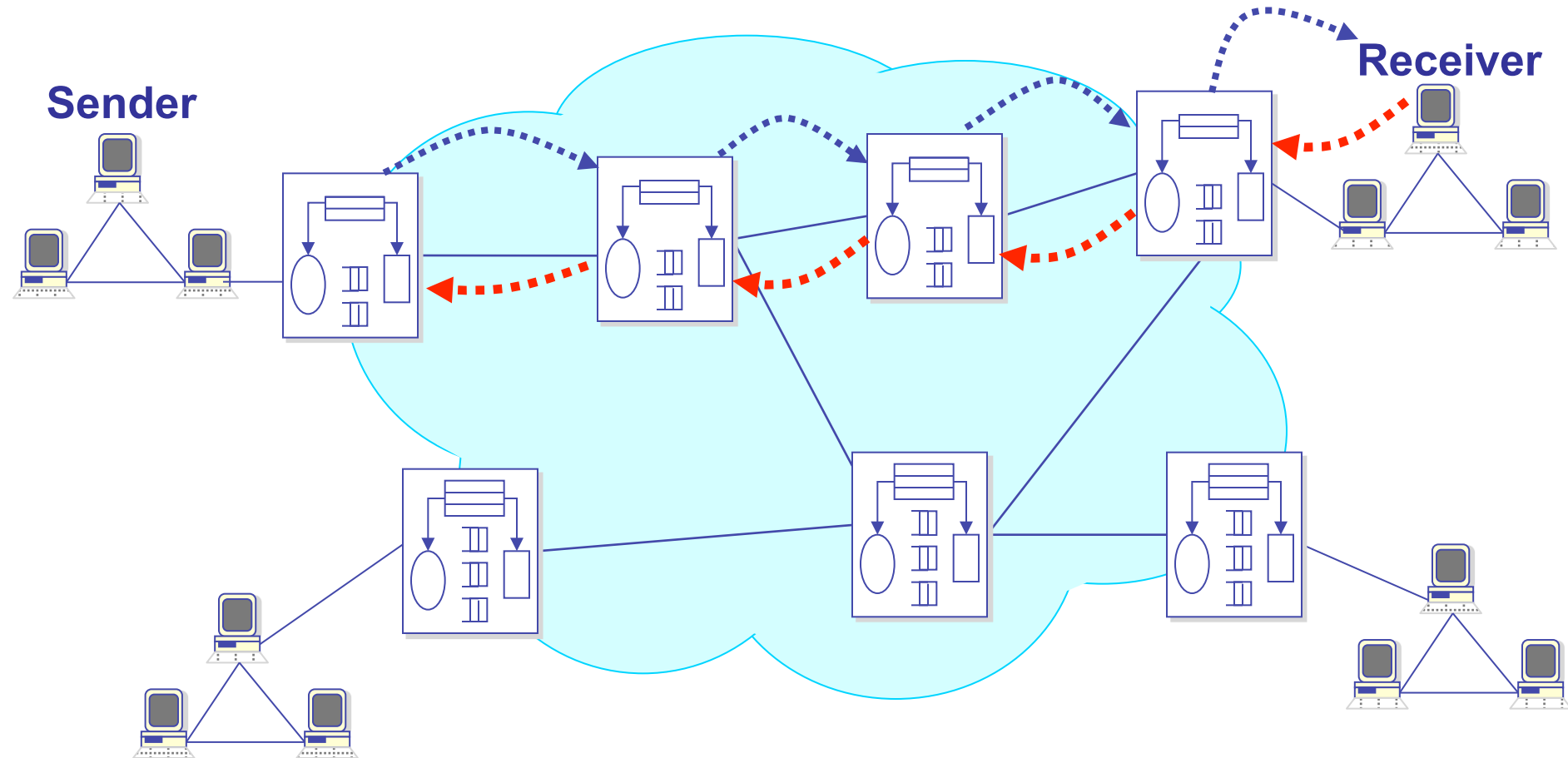
**Receiver**





# Integrated Services

- Allocate resources - perform per-flow admission control



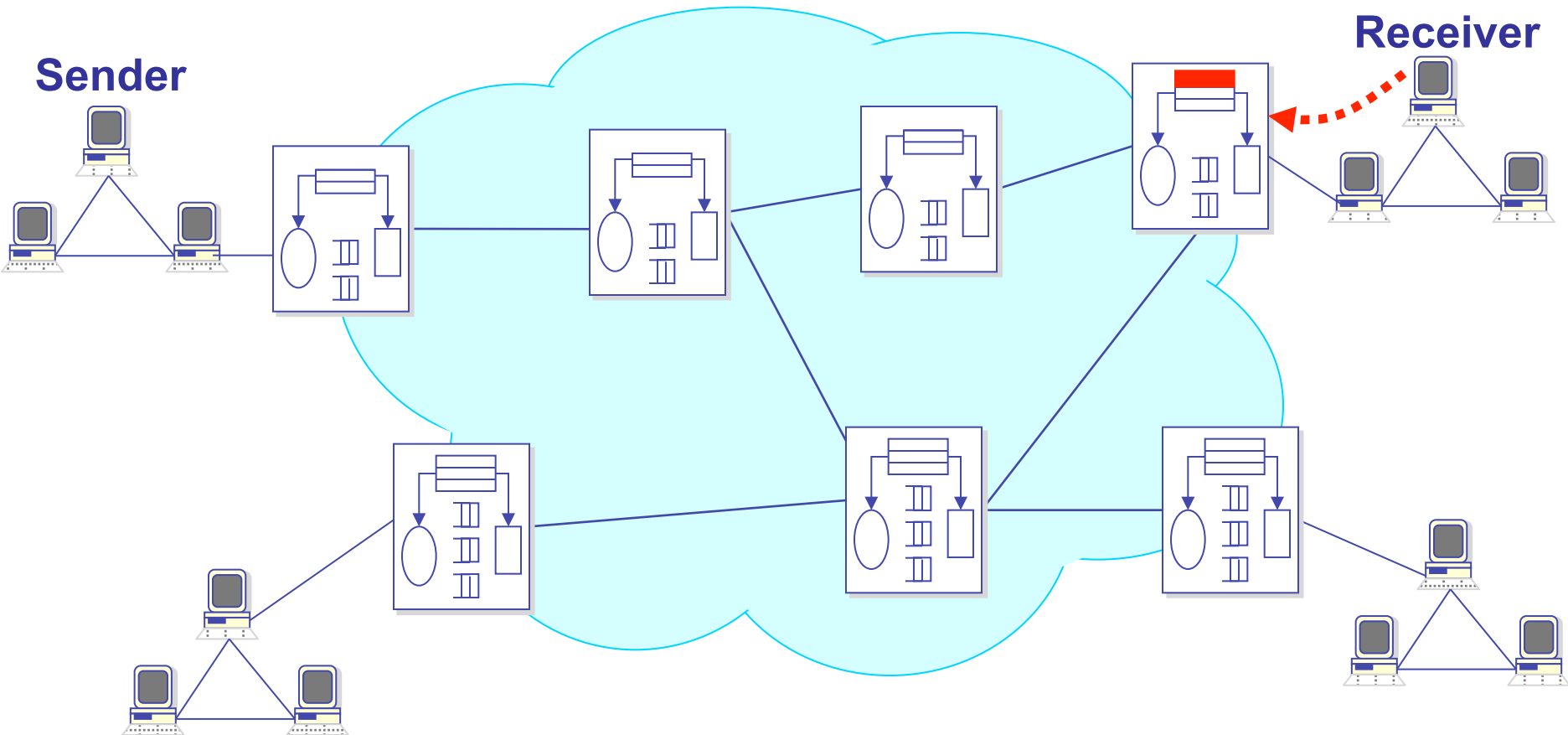


# Integrated Services

- Install per-flow state

**Sender**

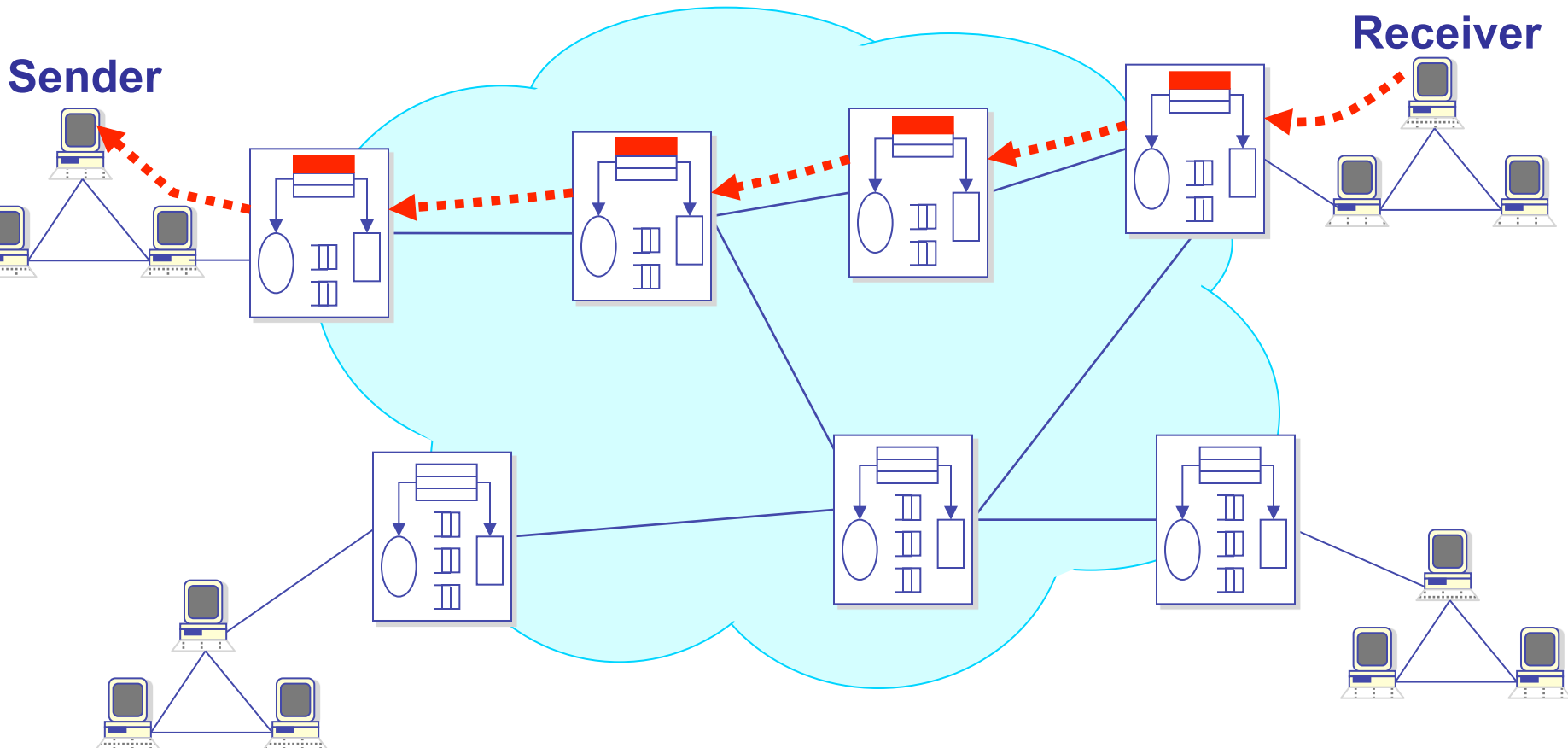
**Receiver**





# Integrated Services

- Install per flow state



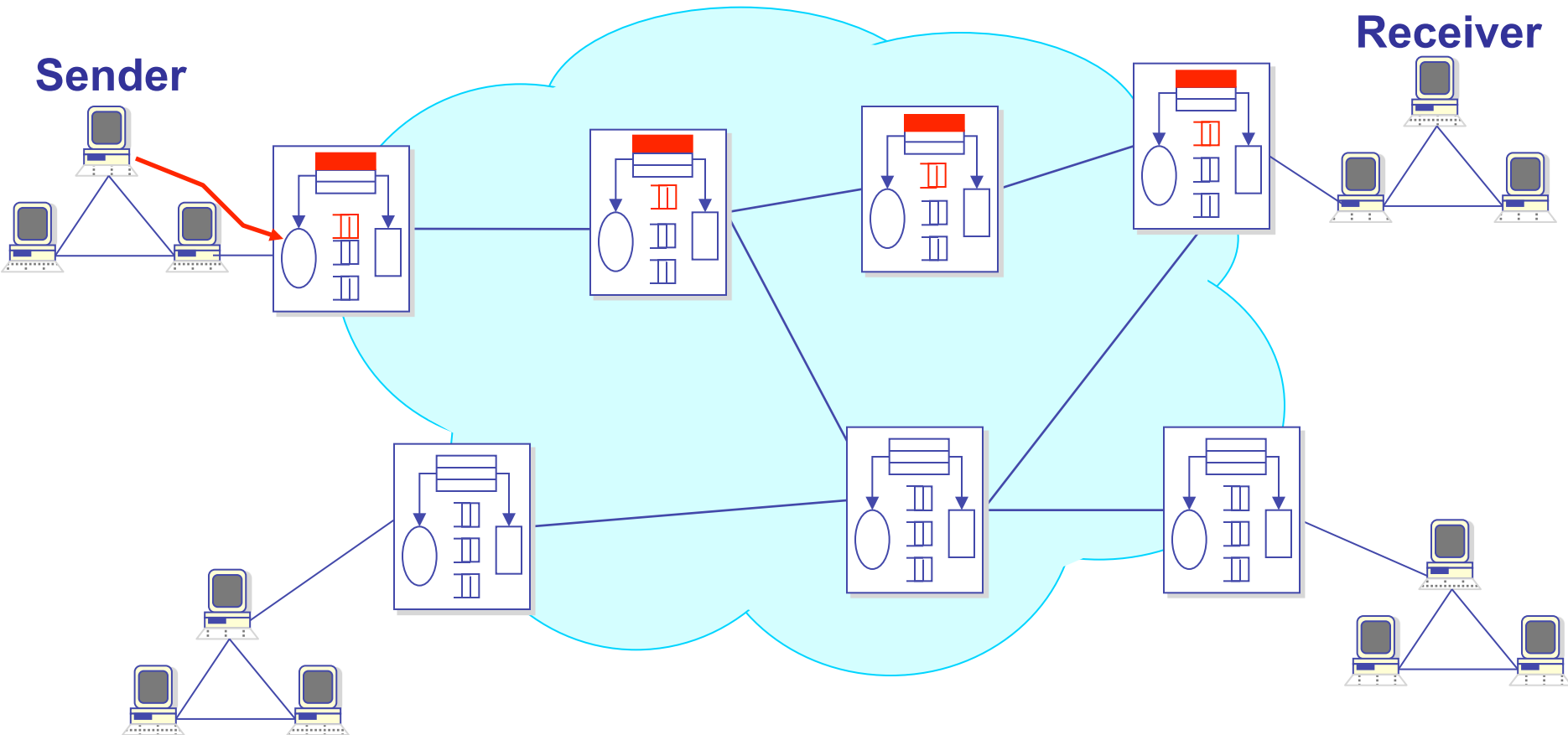


# IntServe: Data Path

- Per-flow classification

**Sender**

**Receiver**



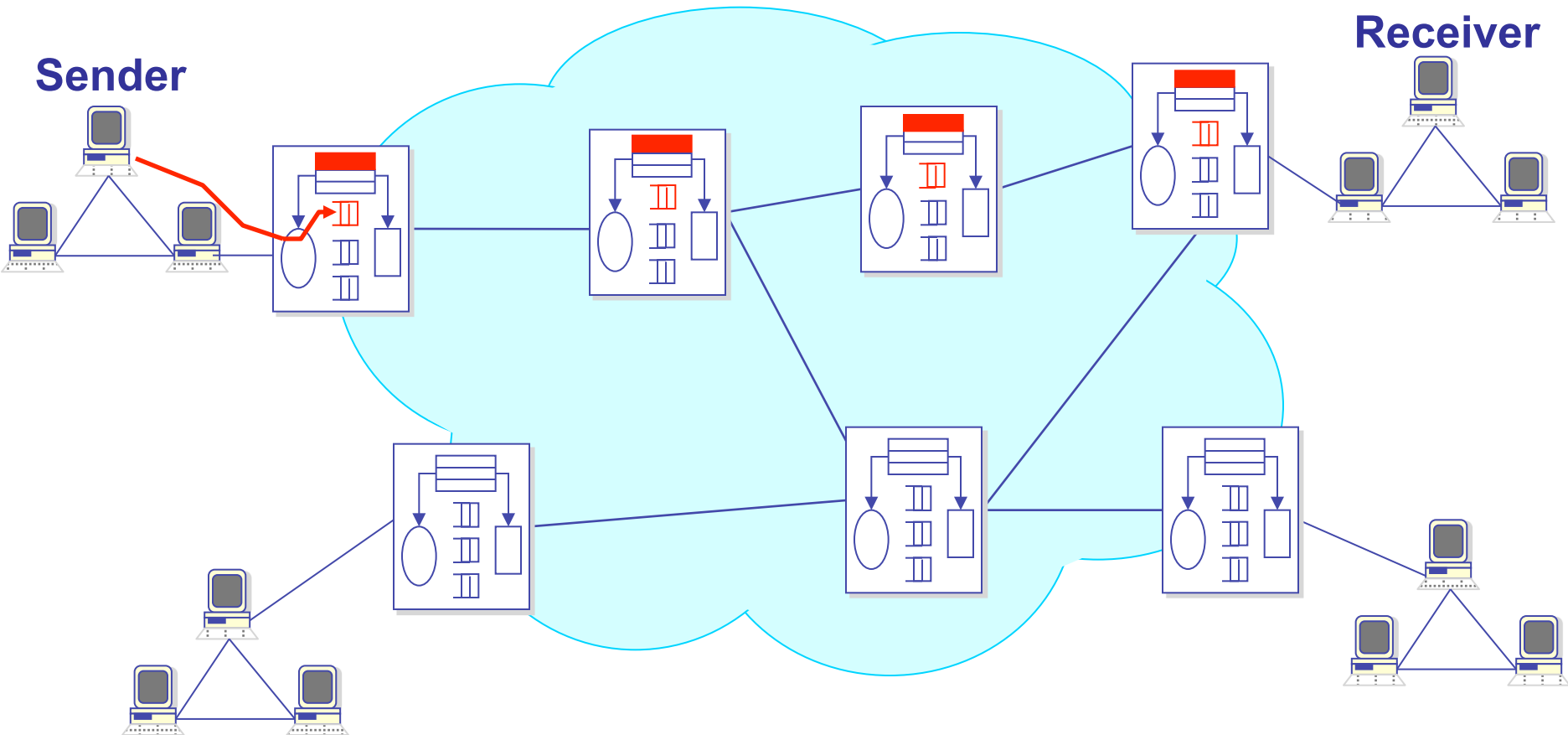


# IntServe: Data Path

- Per-flow buffer management

**Sender**

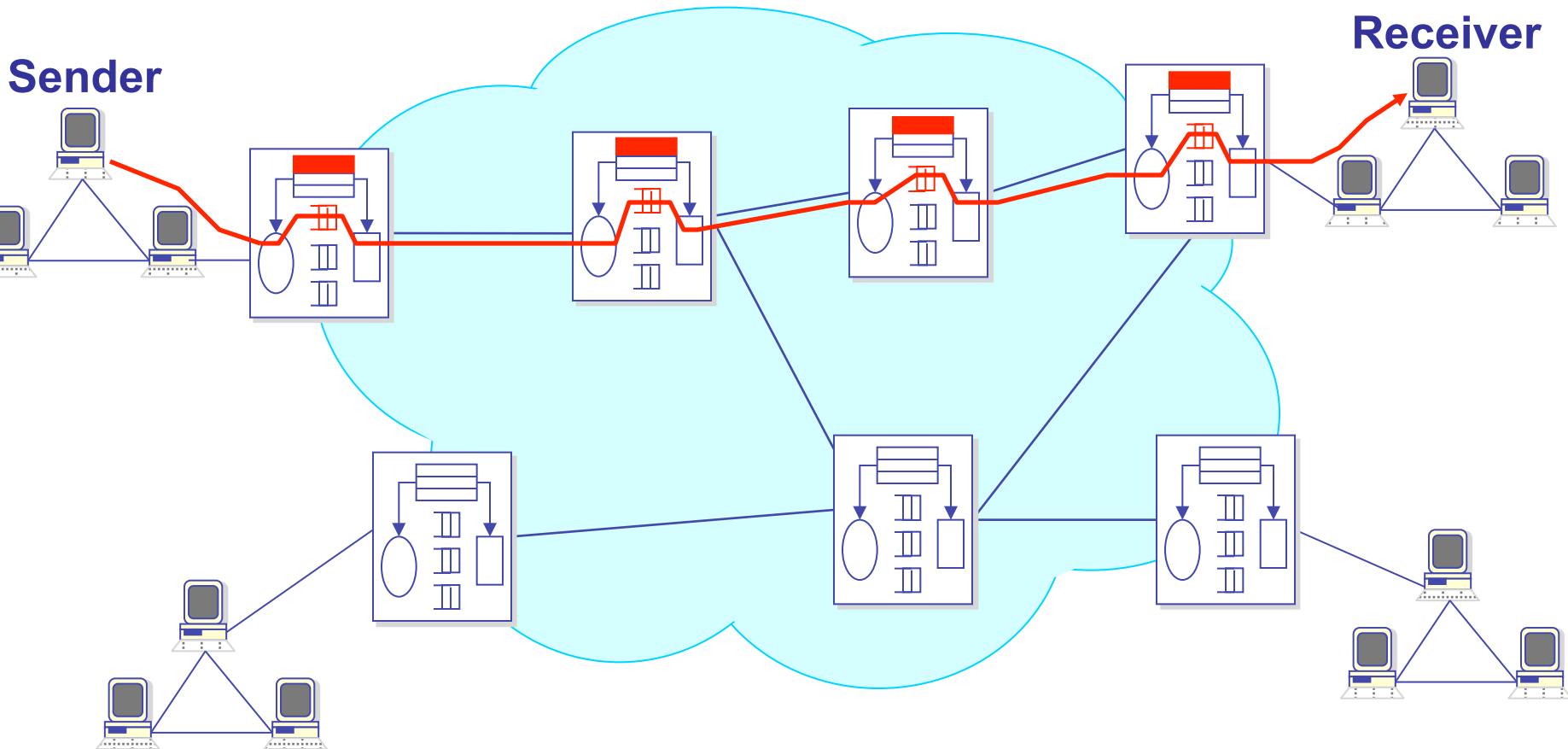
**Receiver**





# IntServe: Data Path

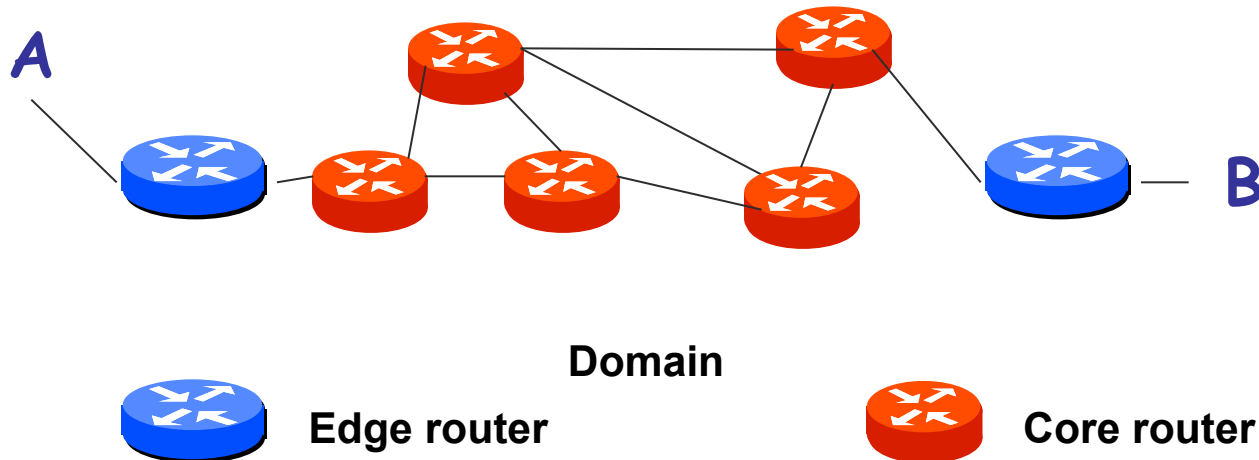
- Per-flow scheduling





# Differentiated Services

- **Edge router**
  - Shape & police traffic
  - Mark “class” of traffic in DS header field (e.g., gold service)
  - Note: DS is taken from the old IP ToS field
- **Core router**
  - Schedule aggregates according to marks in header
  - Drop lower-class traffic first during congestion



# Summary

---



- Routers manage their own resources
  - ◆ Buffer management may entail marking/dropping
  - ◆ Scheduling discipline determines outgoing packet order
  
- Token bucket and RED
  - ◆ Mechanisms to control traffic flowing through routers
  
- Networks can provide quality of service
  - ◆ Combines per-router traffic policing with network signaling
  - ◆ IntServ and DiffServ are contrasting approaches



# For next time...

---

- Read Ch. 2.7 in P&D
- HW 4 due!
- Wrap up Project 2!