Lecture 16:
Quality of Service

CSE 123: Computer Networks

Stefan Savage

Final

- Next week (trust Blink wrt time/location)
- Will cover entire class
- Style similar to midterm
- I’ll post a sample (i.e. old) final tmw

Quality of Service (QoS)

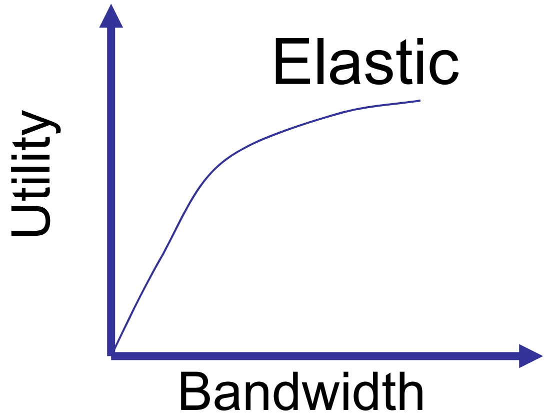
- So far, we have assumed all traffic is equal and provided best effort delivery

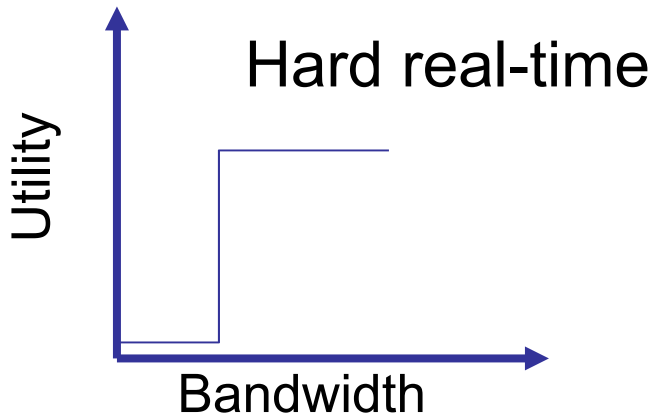
- Not always best model. Why?
  - Application demands
    - I want low-delay low-loss for phone service
  - Market differentiation
    - I want to sell better service for more money
  - Bandwidth management
    - Don't let BitTorrent eat up all UCSD bandwidth
    - Inconsistent TCP implementations (fairness)

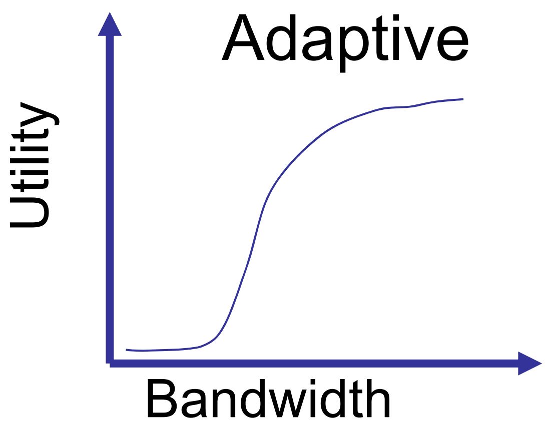
Multimedia Applications

- 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?

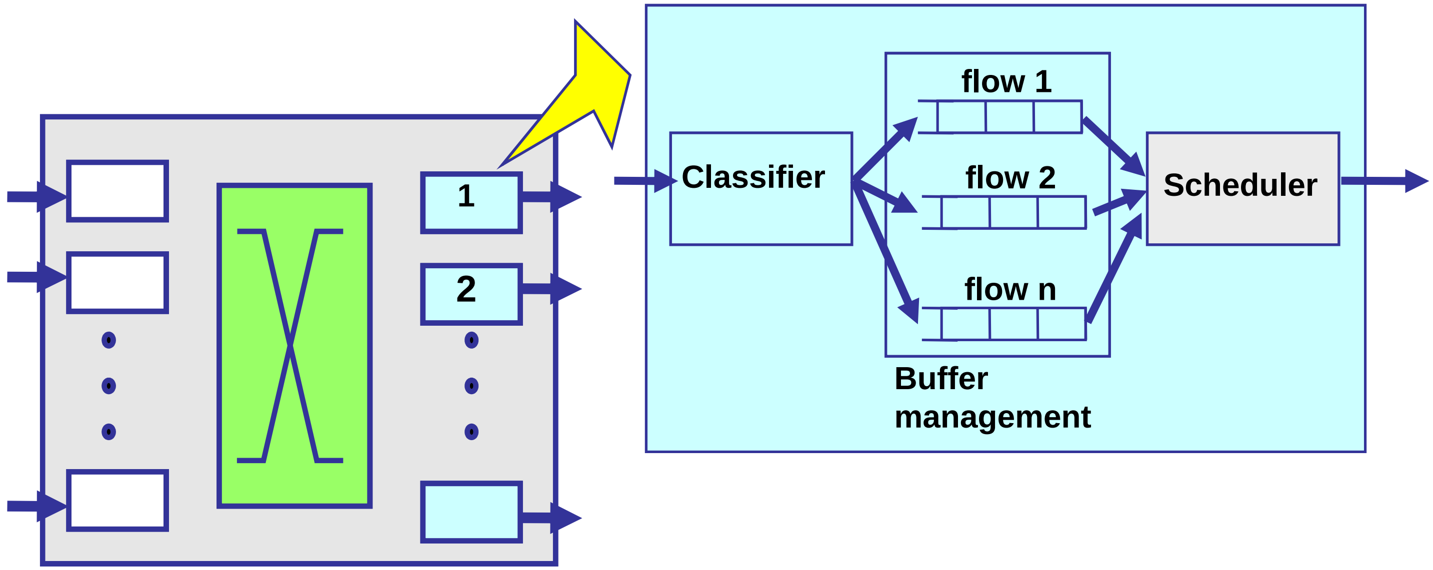
- **Router** classifies based on packet header
  - Aggregates
    - From particular network (IP src address)
    - For particular protocol (e.g., port 80 traffic)
  - Individual network flows
    - 5-tuple (src, dst, src port, dst port, protocol)
  - Special header field that indicates traffic "class"

Possible Service Classes

- Best-effort
  - Vanilla IP
- Differentiated services
  - 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

What tools does router have to implement this? (per link)

- Buffer management: which packet to drop when?
  - We only have finite-length queues
- Scheduling: which packet to transmit next?



Default scheduling/buffer mgmt

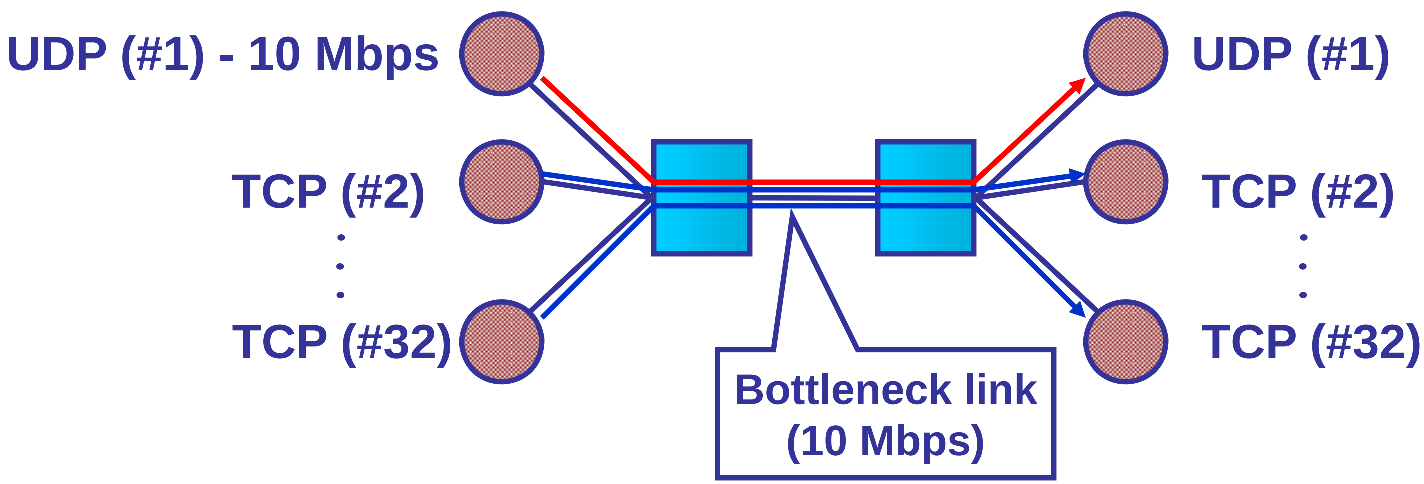
- FIFO + drop-tail
  - Simplest choice
  - Used widely in the Internet
- Important distinction:
  - FIFO: scheduling discipline
  - Drop-tail: drop policy
- FIFO scheduling (first-in-first-out)
  - Implies single class of traffic
- Drop-tail buffer management
  - Arriving packets get dropped when queue is full regardless of flow or importance

FIFO/Drop-Tail Problems

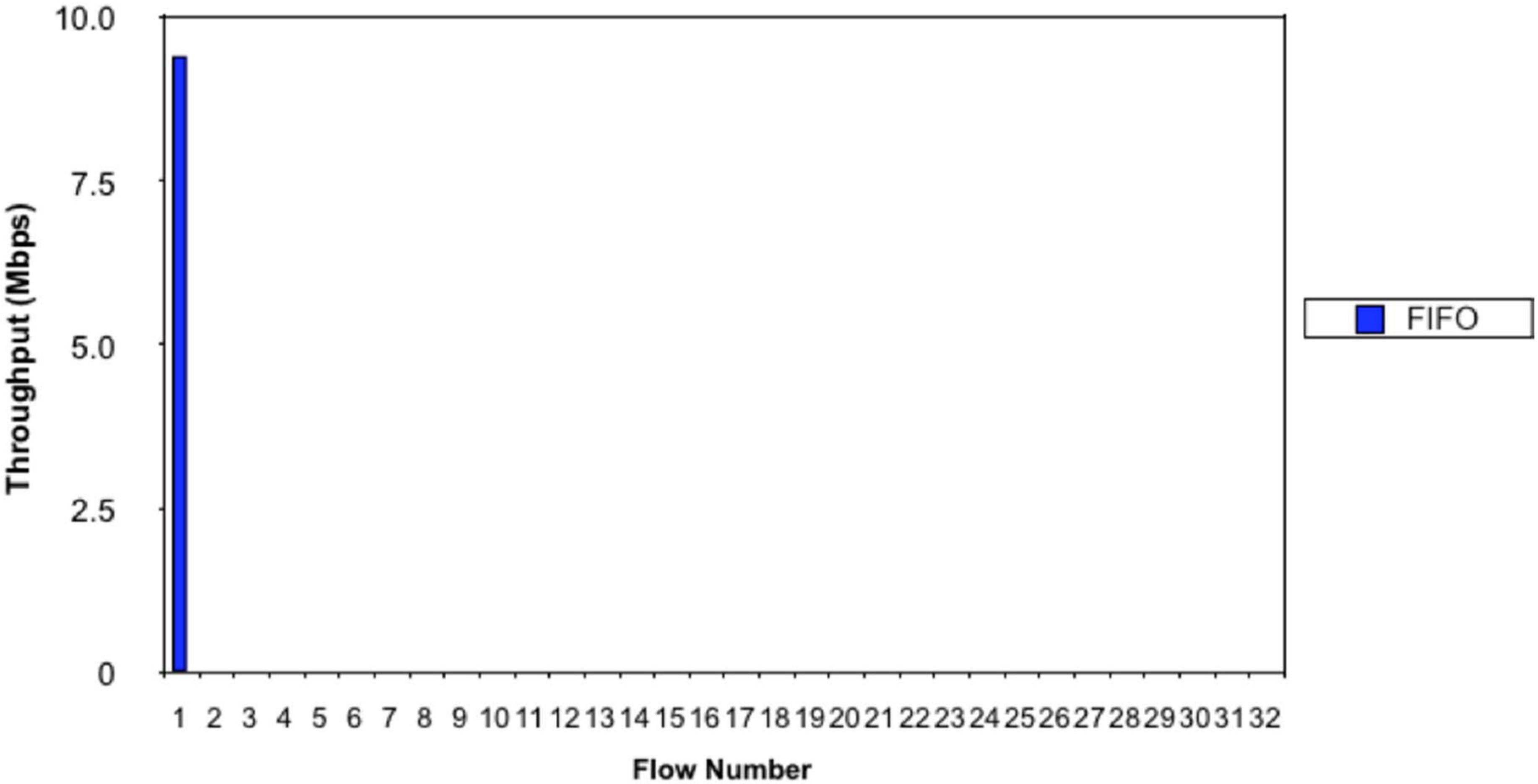
- Leaves responsibility of congestion control completely to the edges (e.g., TCP)
- Does not separate between different flows
- No policing: send more packets → get more service
- Synchronization: end hosts react to same events at same time

Non-responsive Senders

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



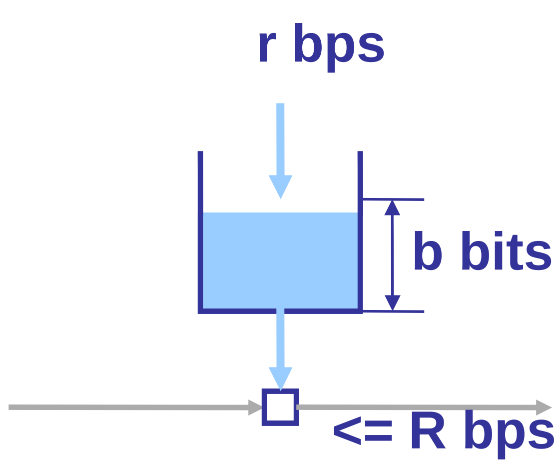
UDP *vs.* TCP

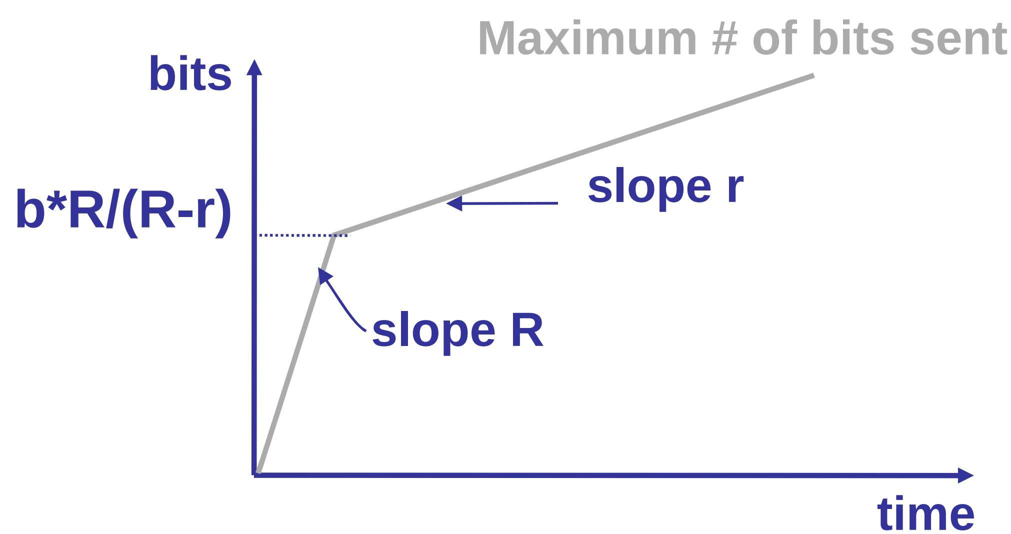


10.0
7.5
5.0
2.5
0
Throughput (Mbps)
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
Flow Number
FIFO

Token Bucket Basics

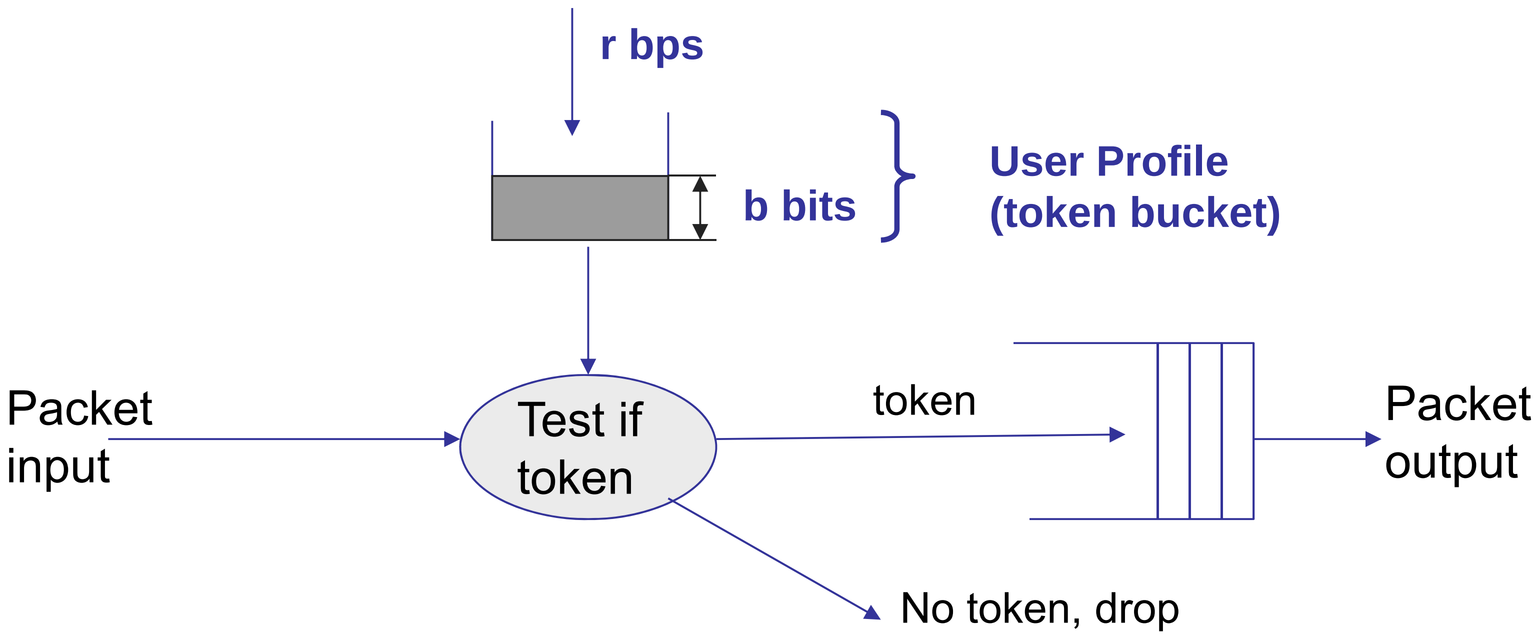
- Parameters
  - r – average rate, i.e., rate at which tokens fill the bucket
  - b – bucket depth
  - R – maximum link capacity or peak rate (optional parameter)
- A bit is transmitted only when there is an available token





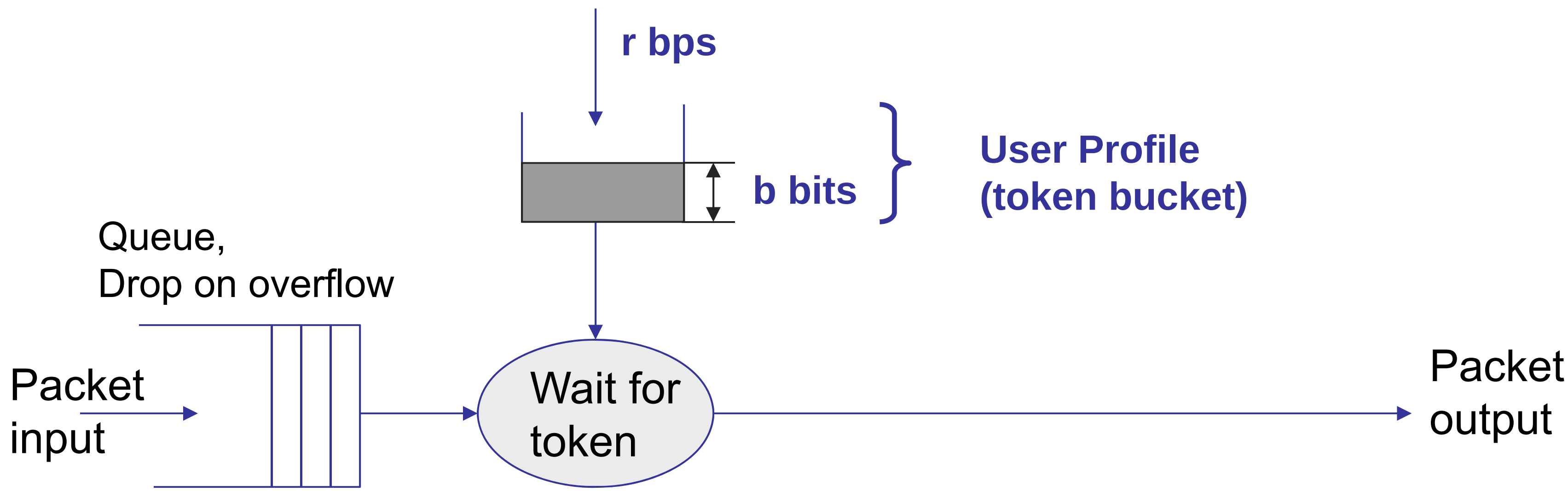
Traffic Policing

- Drop packets that don't meet user profile
- Output limited to average of *r* bps and bursts of *b*



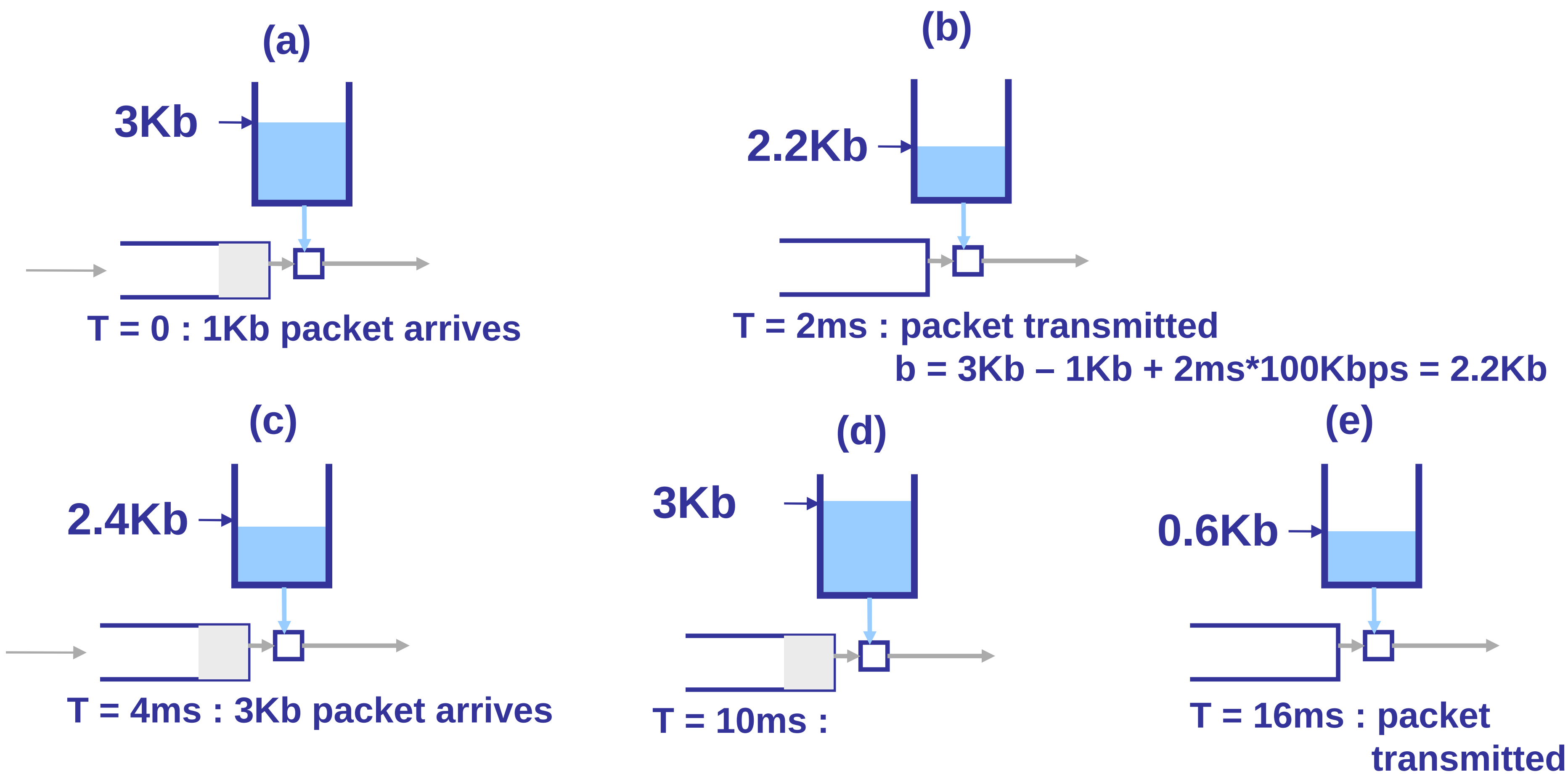
Traffic Shaping

- Shape packets according to user profile
- Output limited to average of *r* bps and bursts of *b*



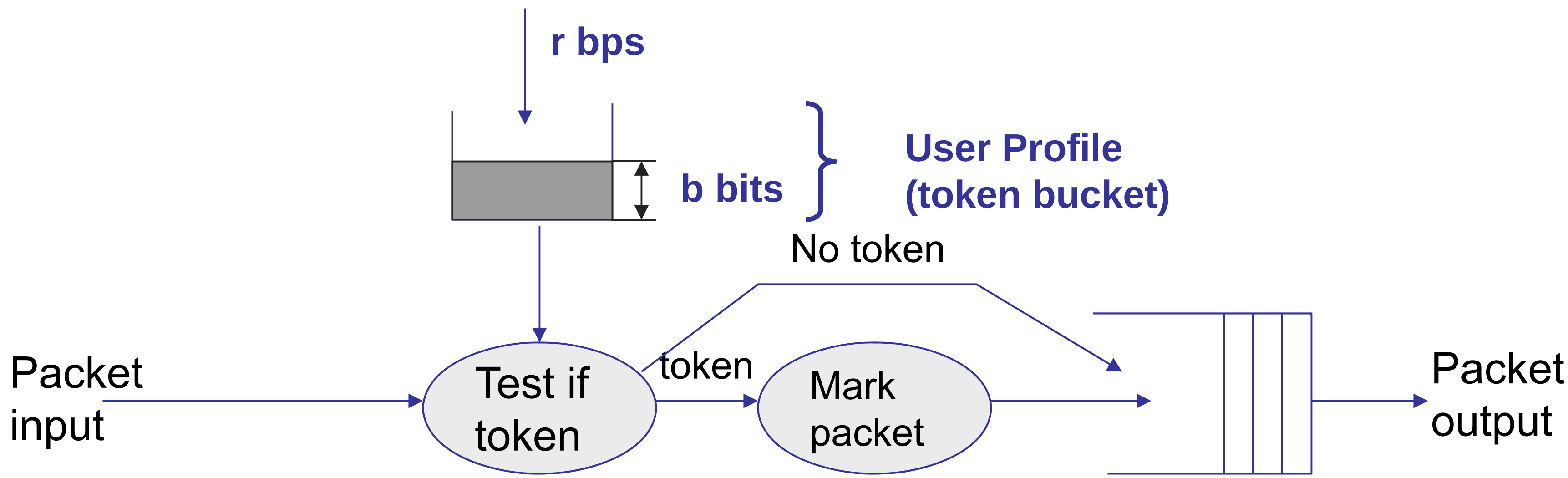
Shaping Example

- r = 100 Kbps; b = 3 Kb; R = 500 Kbps

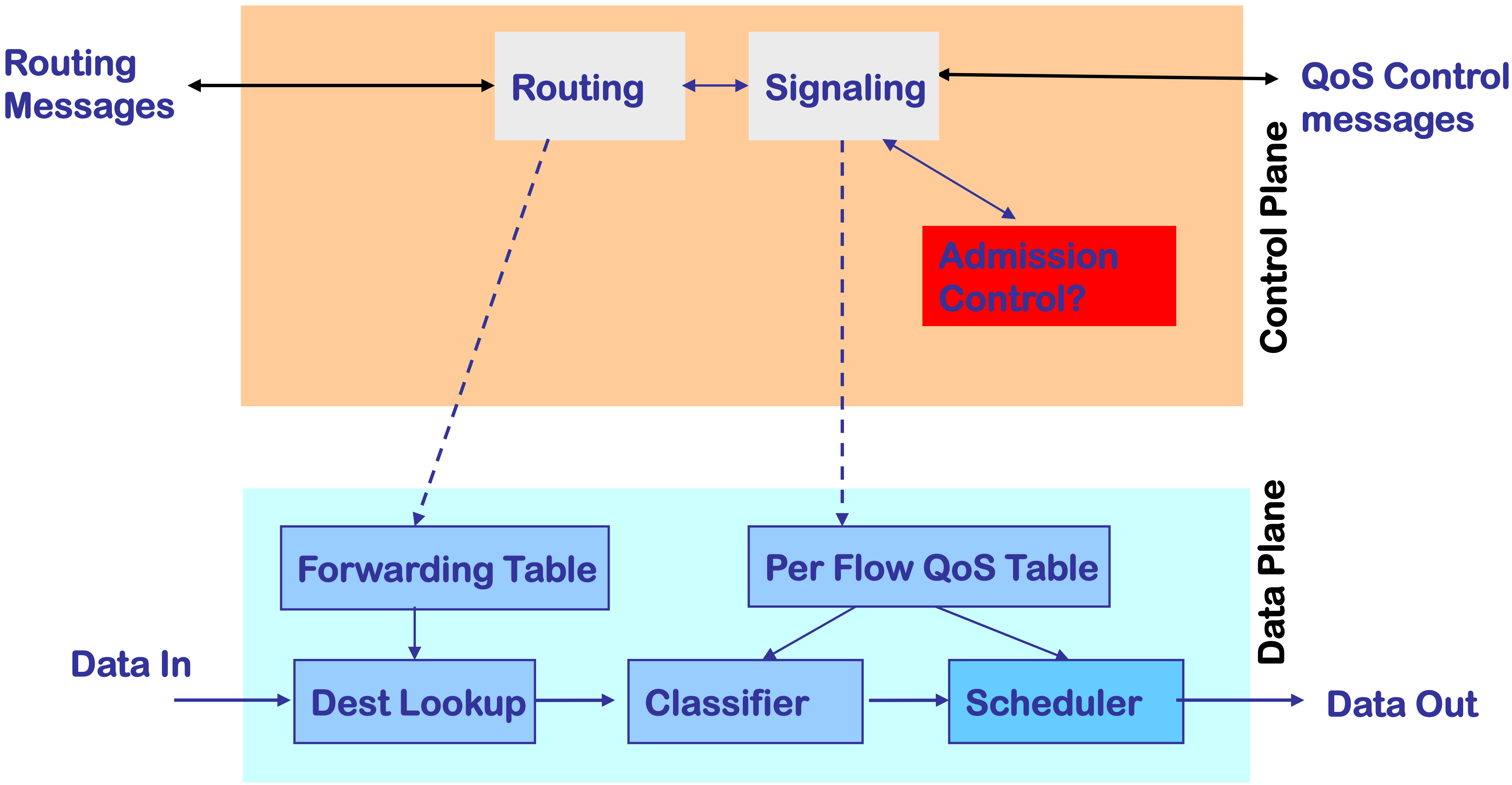


Buffer Management

- Mark packets according to flow's token bucket profile
- During congestion, drop unmarked pkts first



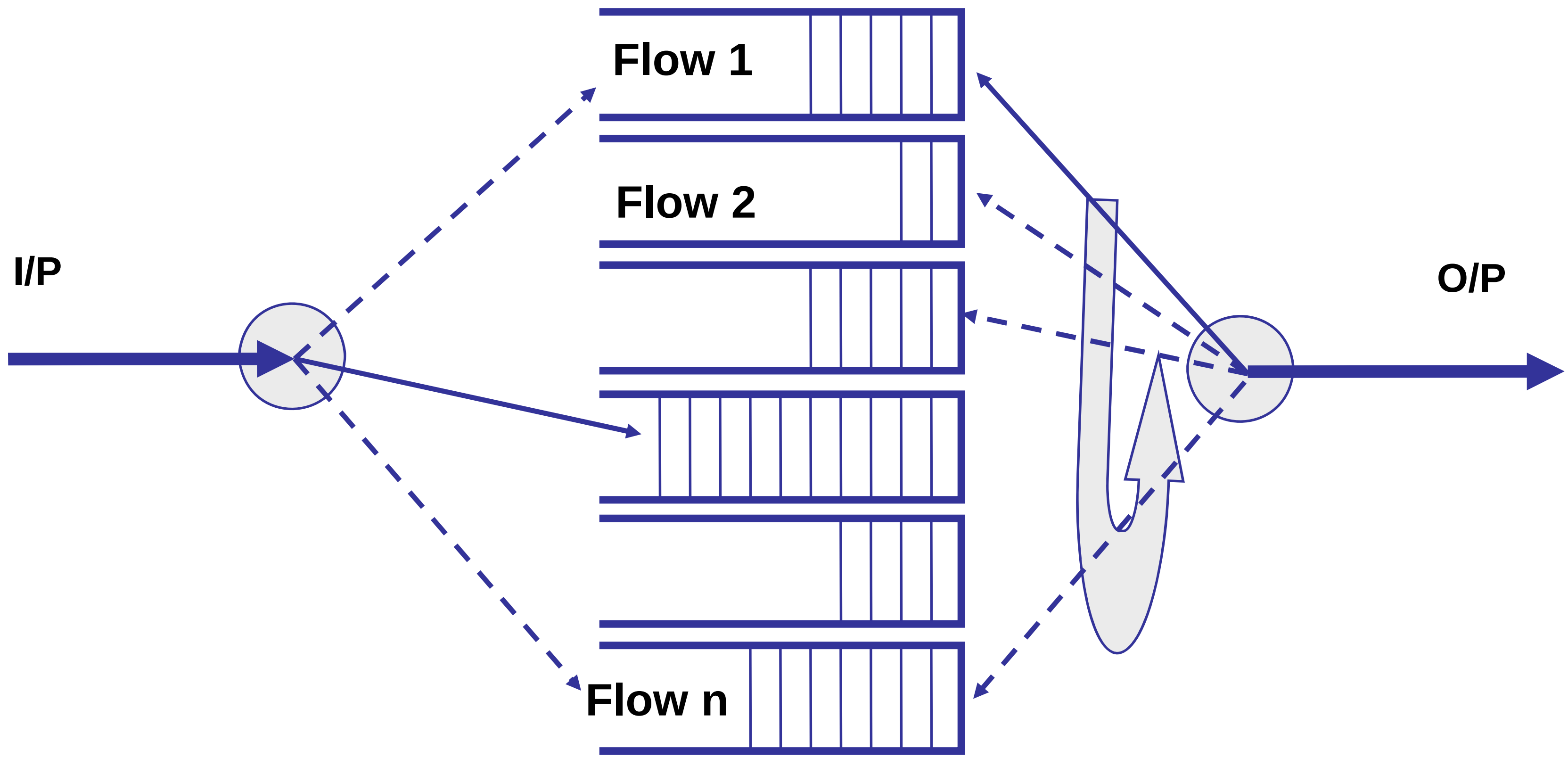
More Complicated Routers



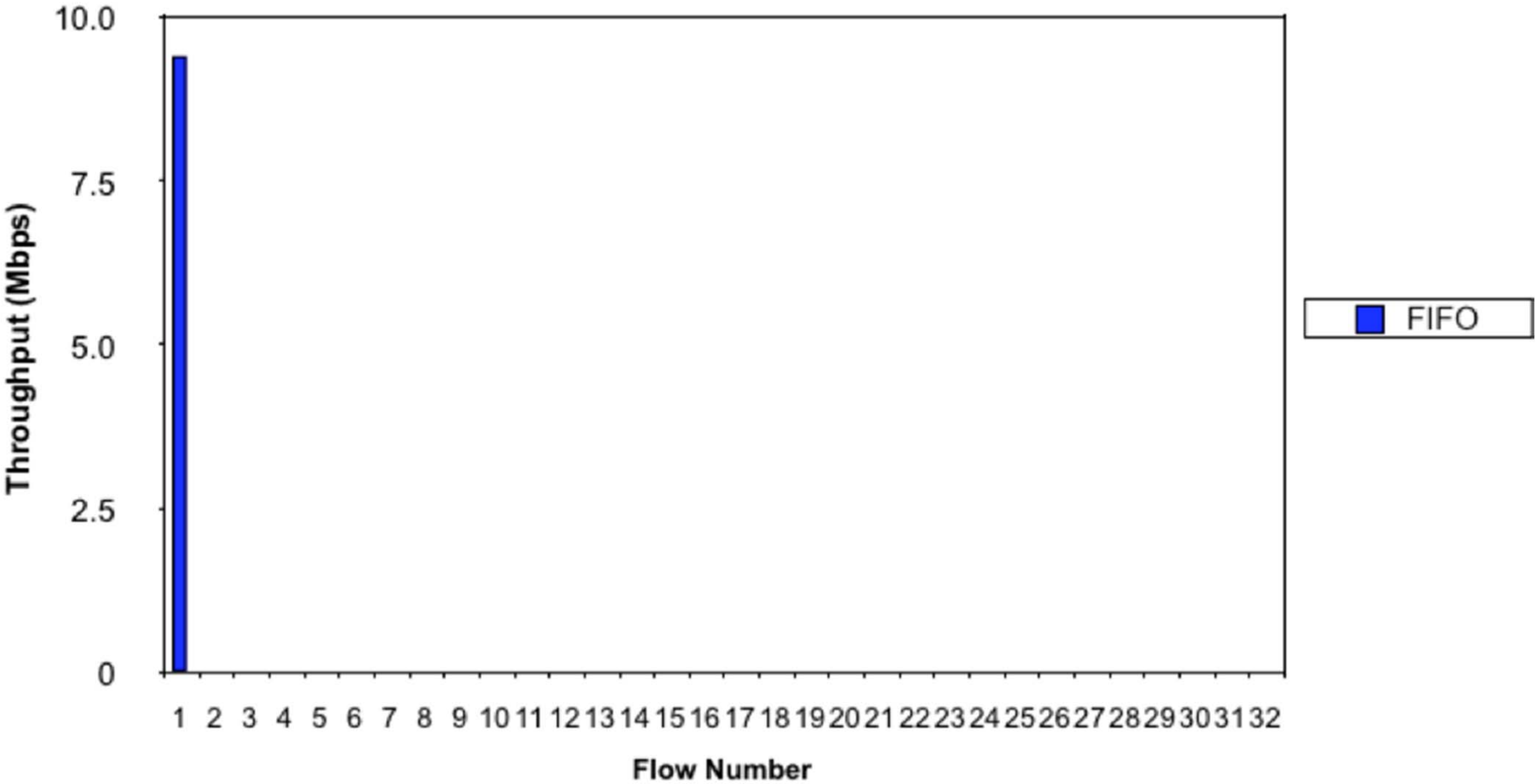
Scheduling

- So far we've looked at flow-based traffic policing
  - Limit the rate of one flow regardless 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)
- Fair queuing
  - Proportional share scheduling
  - Schedule round-robins among queues
  - Weighted FQ: schedules in proportion to some weight parameter

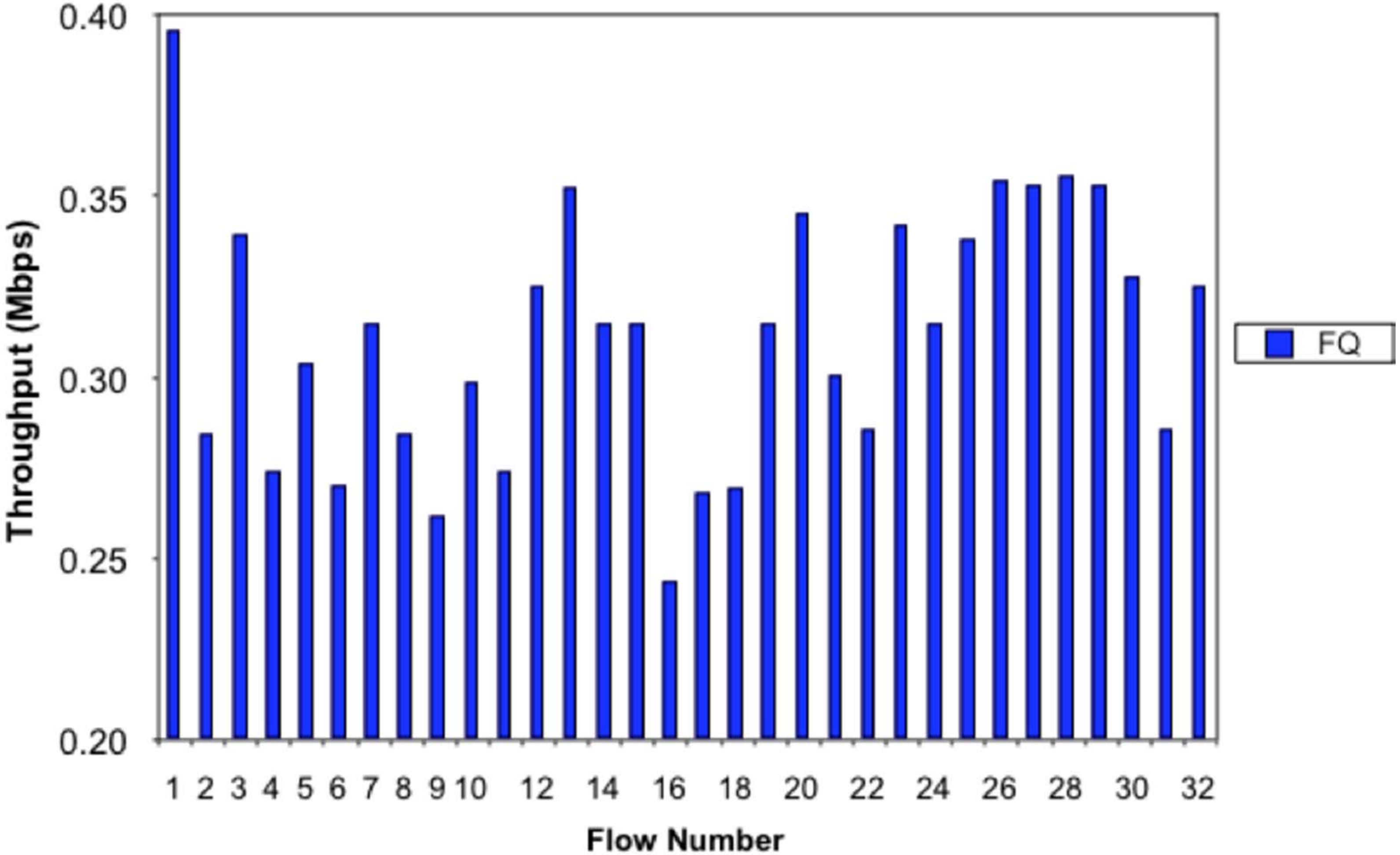
(Weighted) Fair Queuing



UDP *vs.* TCP wo/Fair Queuing



TCP vs. UDP w/Fair Queuing



Fair Queuing

- Maintain a queue for each flow
  - What is a flow?

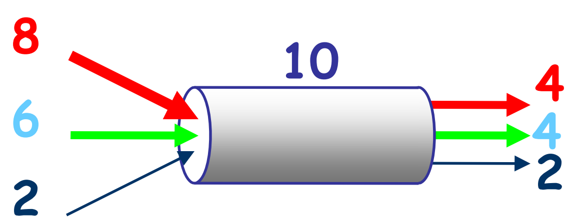
- Implements max-min fairness: each flow receives $min(r_i, f)$, where
  - r_i – flow arrival rate
  - f – link fair rate (see next slide)

- Weighted Fair Queuing (WFQ) – associate a weight with each flow

Fair Rate Computation

- If link congested, compute f such that

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

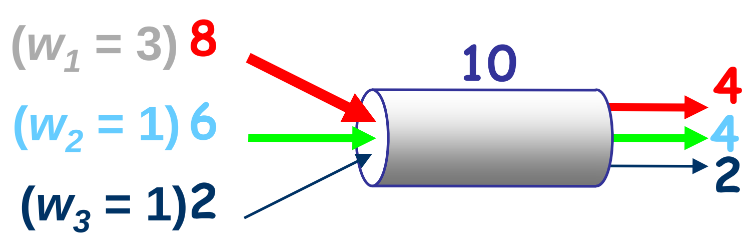


$f = 4$:
min(8, 4) = 4
min(6, 4) = 4
min(2, 4) = 2

Another Example

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

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



f = 2:
min(8, 2*3) = 6
min(6, 2*1) = 2
min(2, 2*1) = 2

Flow i is guaranteed to be allocated a rate >= w_i*C/(Σ_k w_k)

⇓

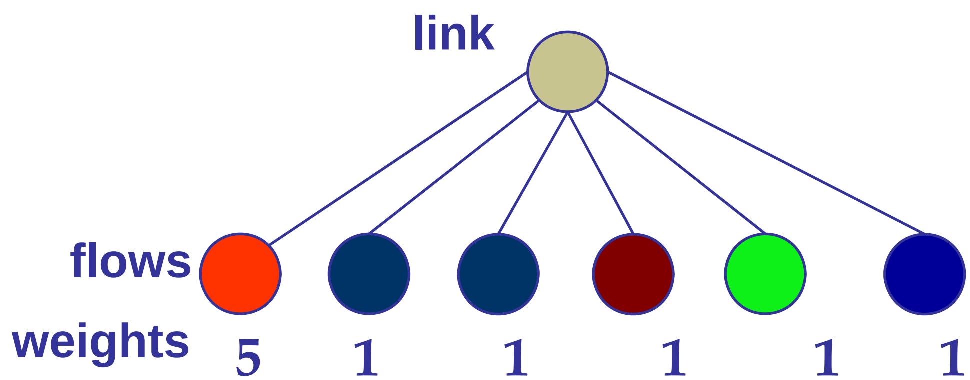
If Σ_k w_k <= C, flow i is guaranteed to be allocated a rate >= w_i

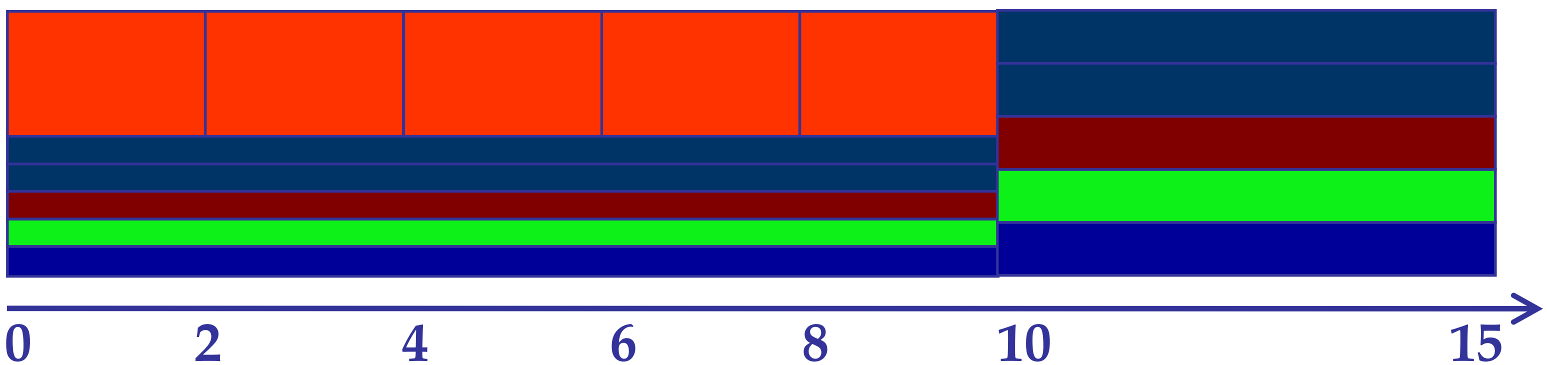
Fluid Flow model

- 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



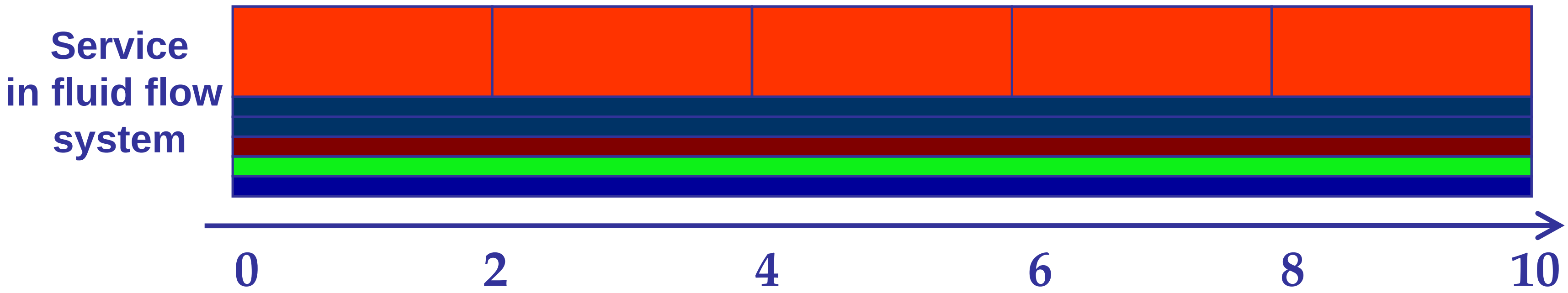


Packet-Based Implementation

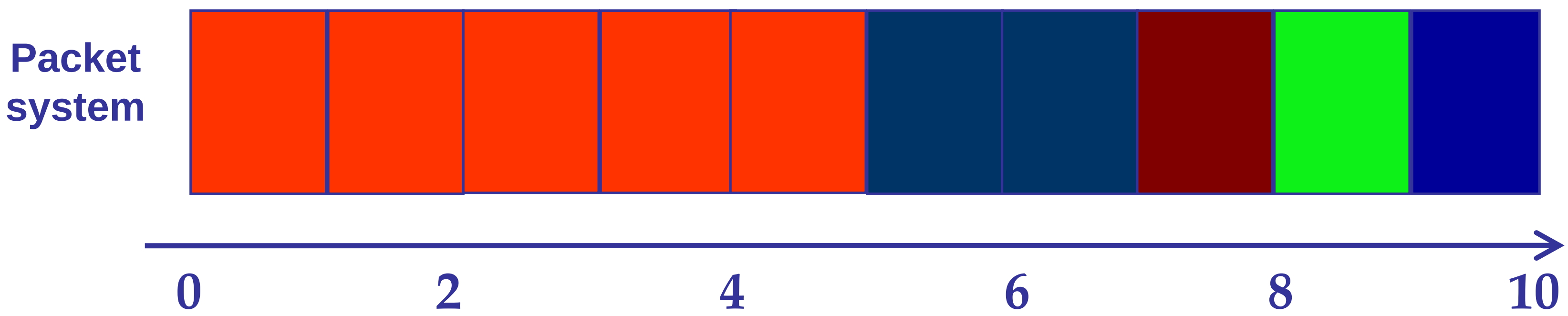
- Packet (Real) system: packet transmission cannot be preempted. Why?

- 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



Network-wide QoS

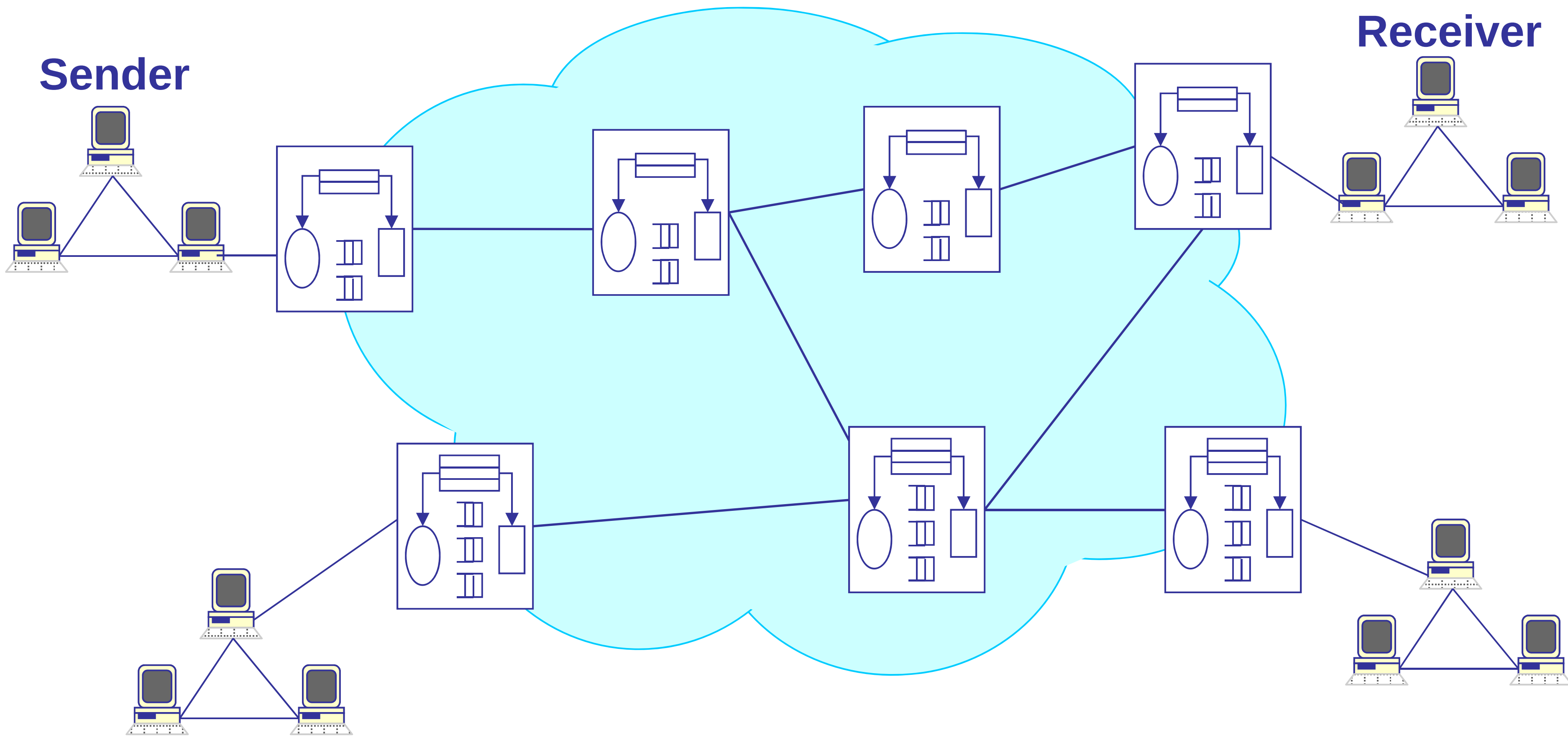
- Integrated services
  - Motivated by need for end-to-end guarantees
  - On-line negotiation of per-flow requirements
  - End-to-end per-router negotiation of resources
  - Complex
- Differentiated services
  - Motivated by economics (multi-tier pricing)
  - No per-flow state
  - Not end-to-end and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

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 to deliver

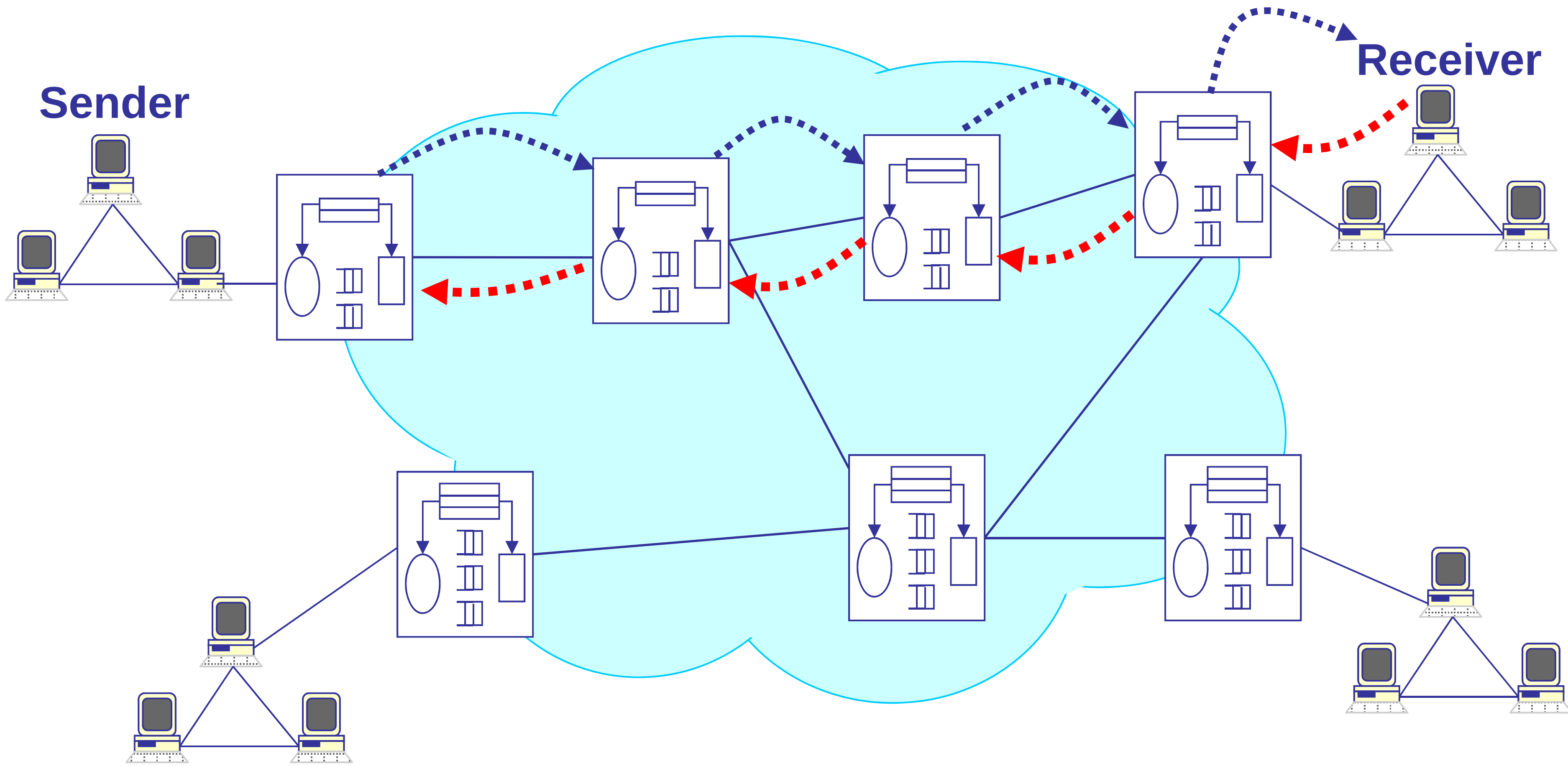
Integrated Services

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



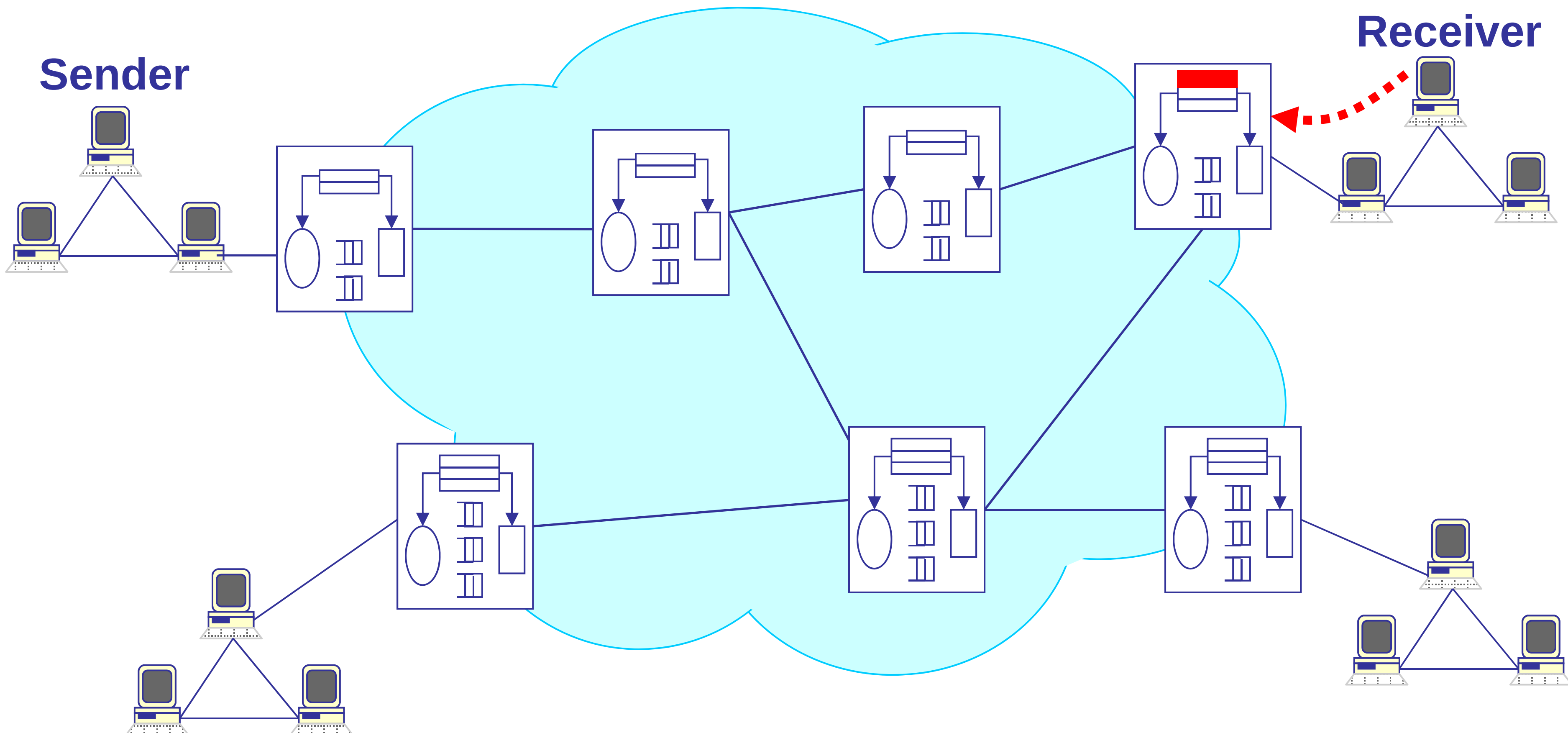
Integrated Services

- Allocate resources - perform per-flow admission control



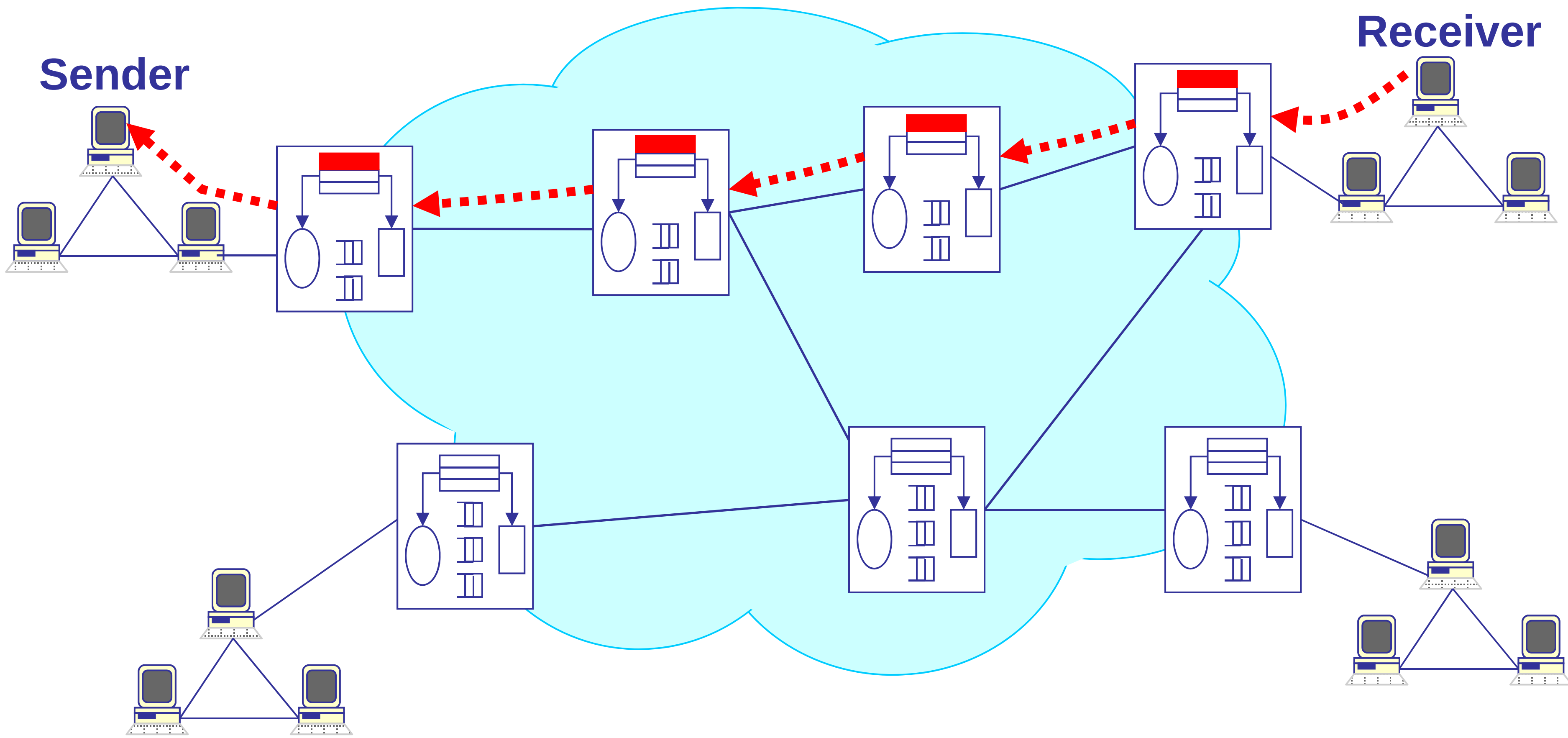
Integrated Services

- Install per-flow state



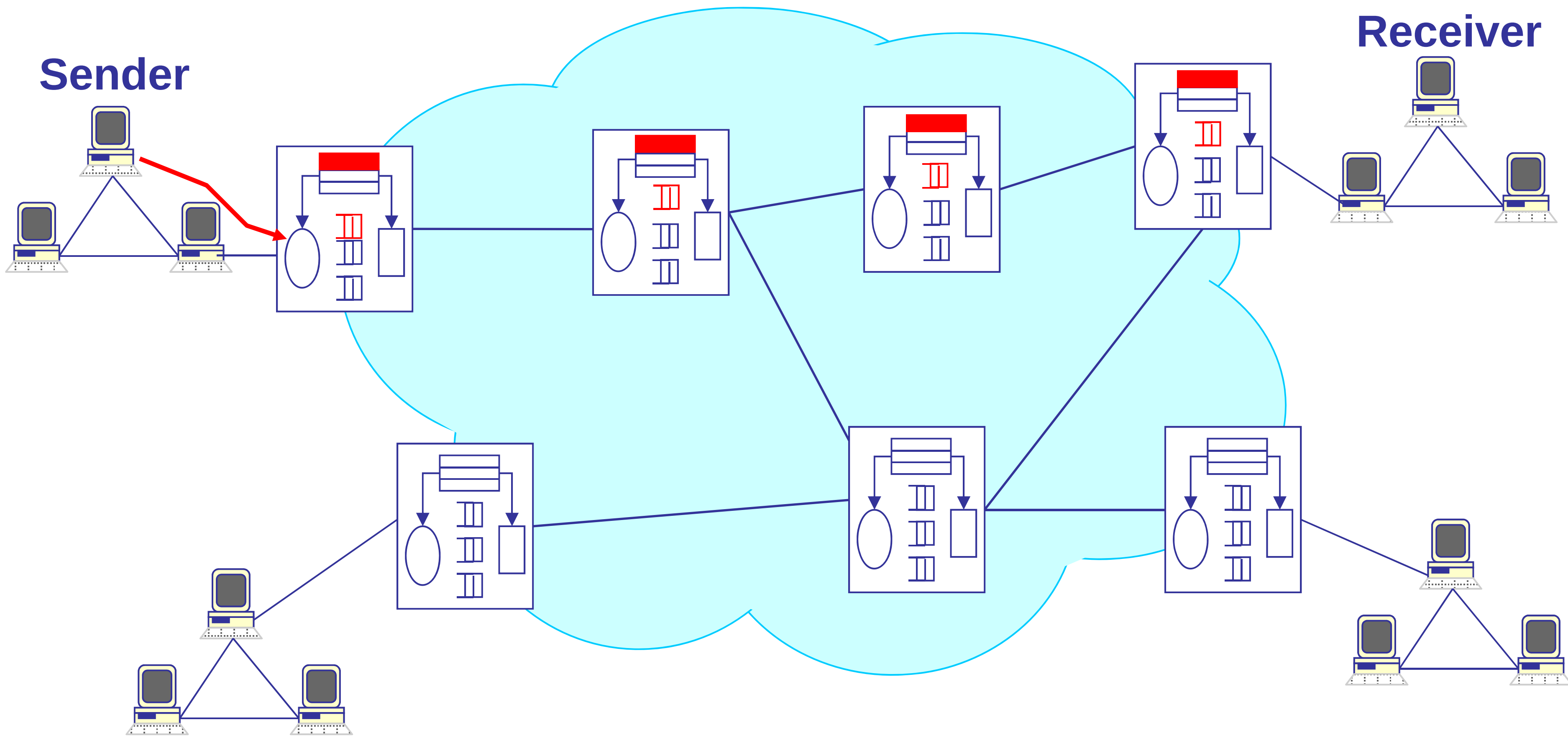
Integrated Services

- Install per flow state



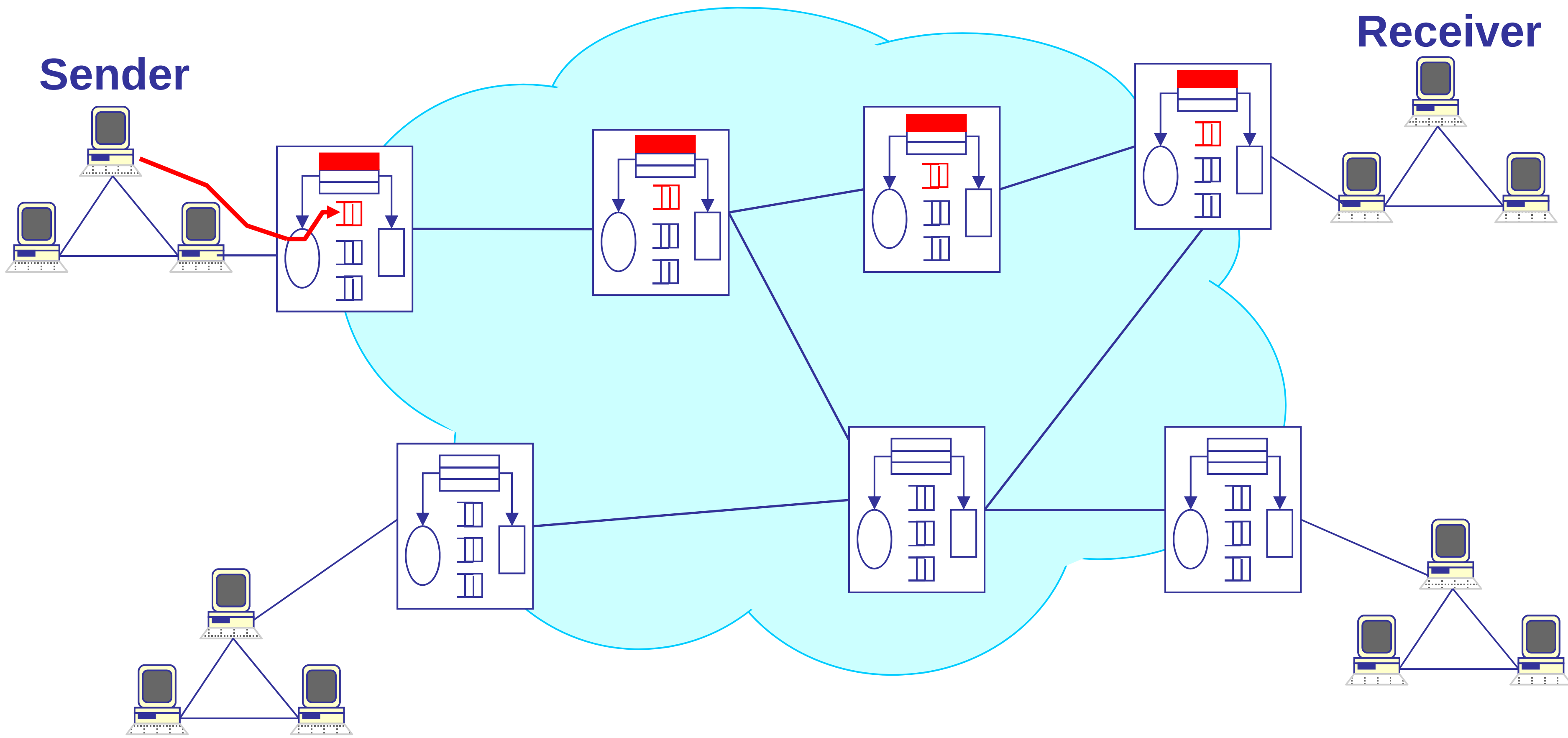
IntServe: Data Path

- Per-flow classification



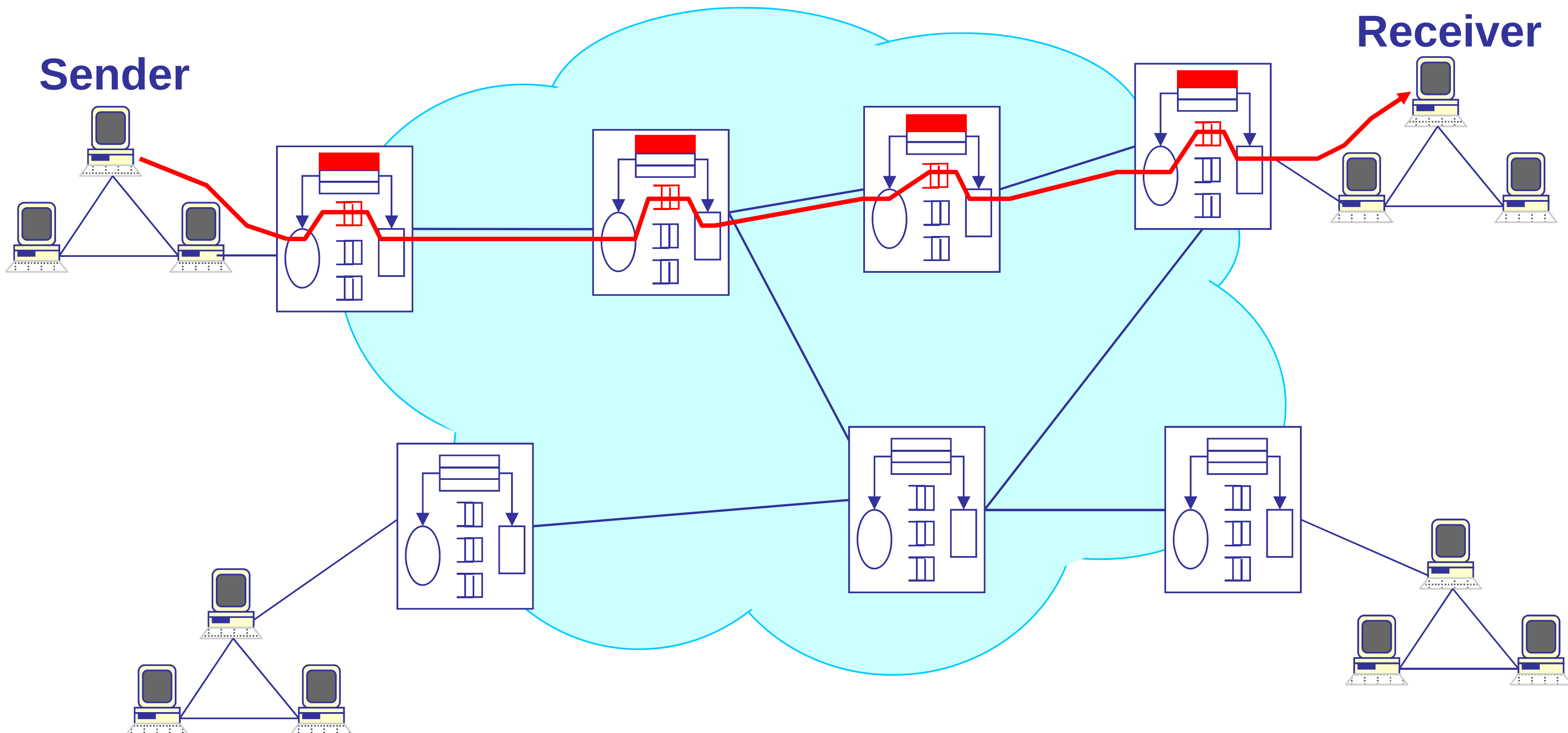
IntServe: Data Path

- Per-flow buffer management



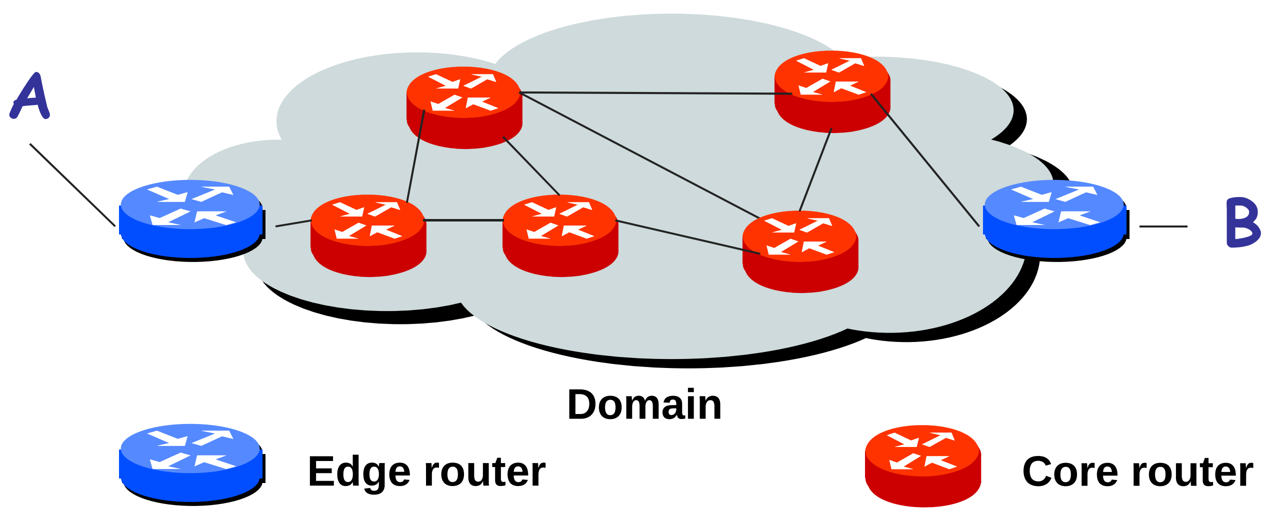
IntServe: Data Path

- Per-flow scheduling



Differentiated Services

- Edge router
  - Shape & police traffic
  - Mark “class” of traffic in IP header field (e.g., gold service)
- Core router
  - Schedule aggregates according to marks in header
  - Drop lower-class traffic first during congestion



Summary

- To enforce differentiation on traffic quality requires router support
  - Buffer management: what gets dropped
  - Scheduling: what gets sent when
- Token bucket
  - Key abstraction for taking about traffic needs (rate and burstiness)
- Fair queuing
  - Approach to allocate bandwidth between flows
- Networks can provide quality of service
  - Combines per-router traffic policing with network signaling
  - IntServ and DiffServ are contrasting approaches

Thanks!

- See you at the final!