
OSPF, Congestion Control

18 May 2011
Lecture 10

Slides Credits: Steve Zdancewic (UPenn)

Topics for Today

- End to End Example
- OSPF
- Congestion Control
 - Queuing

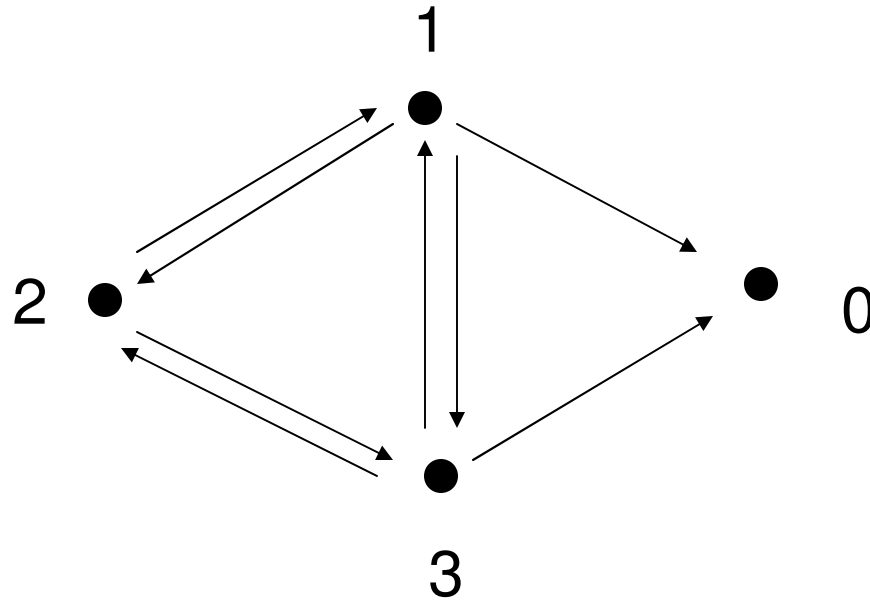
- Sources in PD:
 - OSPF 4.2.3
 - Congestion Control 6.1-6.2

End to End Example

- Example of DHCP, ARP, DNS protocols interacting

RIP Issues: Stability problem

- Loops may form and stability cannot occur without “counting to infinity”.
- Timing of events may cause cycles of updates.



(Partial) Solutions for Stability

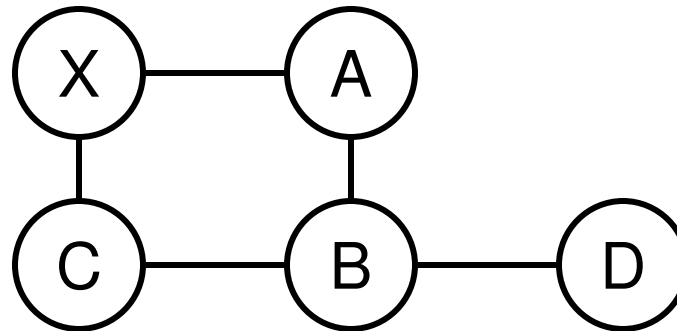
- Pick small value for “infinity”
 - If “infinity” is 16 then at most 15 hops in network
 - Distance of 16 considered unreachable
- Disallow cyclic updates
 - Called *split horizon* algorithm
 - Don't send updates learned from a neighbor back to that neighbor
 - Only works for small (e.g. 2-hop) cycles

Open Shortest Path First (OSPF)

- Each node sends a *reliable flood* of information to all other nodes
- These **Link-State Packets** (LSPs) contain
 - ID of the node that created the packet
 - List of (neighbor, cost) pairs associated with the source node
 - Sequence Number (64bits—no wrapping)
 - Time To Live (ensure old info is eventually removed)

Reliable Flooding

- Transmission between adjacent routers is made reliable through ACKs & retransmission
- Source sends to all neighbors
- Each recipient
 - Sends to all neighbors except the one it got the message from
 - Ignores duplicates



OSPF continued

- Once all of the link-state info has been flooded each node has complete network topology
- Compute routing information using Dijkstra's shortest-path algorithm
- Periodic updates and failure detection are like RIP.

OSPF Features

- Authentication of routing messages
 - Misconfigured or malicious host could advertise bad route info (i.e. reach anywhere in 0 hops)
 - (Eventually added to RIP too.)
- Additional Hierarchy
 - Partitions domains into *areas*
 - Reduces transmission & storage overhead
- Load Balancing
 - Multiple routes with same cost
 - Traffic evenly distributed

Dijkstra's Algorithm

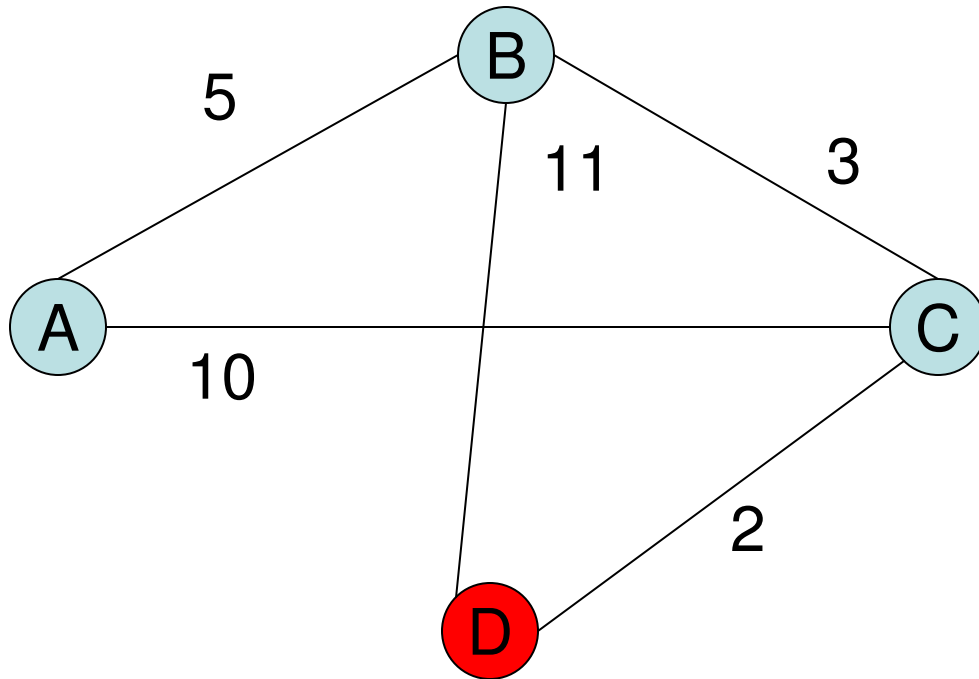
Each node has two lists – Confirmed and Tentative

- Each is pairs of (Destination, Cost, Next-Hop)

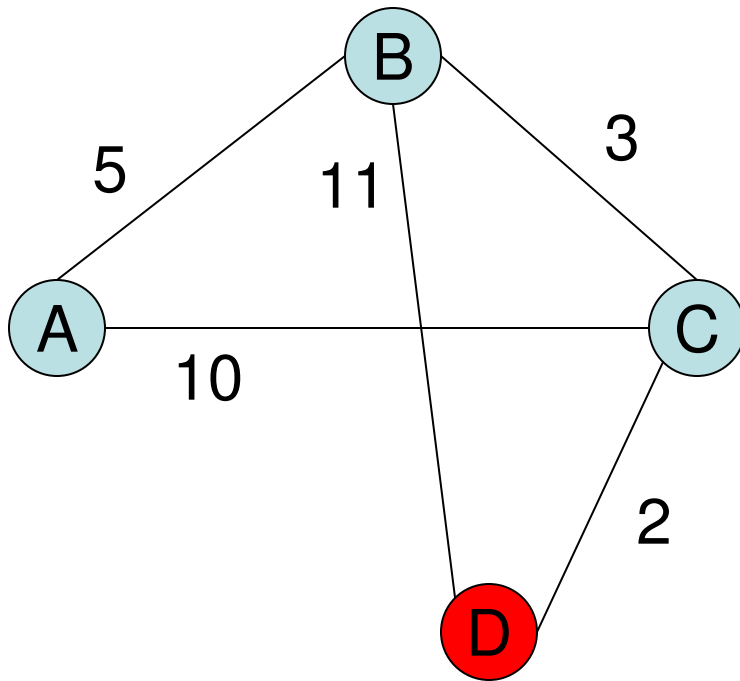
The algorithm:

1. Initialize the Confirmed list with an entry for myself; this entry has a cost of 0.
2. For the node just added to the Confirmed list in the previous step, call it node Next, select its LSP.
3. For each neighbor (Neighbor) of Next, calculate the cost (Cost) to reach this Neighbor as the sum of the cost from myself to Next and from Next to Neighbor.
 - a) If Neighbor is currently on neither the Confirmed nor the Tentative list, then add (Neighbor, Cost, NextHop) to the Tentative list, where NextHop is the direction I go to reach Next.
 - b) If Neighbor is currently on the Tentative list, and the Cost is less than the currently listed cost for Neighbor, then replace the current entry with (Neighbor, Cost, NextHop), where NextHop is the direction I go to reach Next.
4. If the Tentative list is empty, stop. Otherwise, pick the entry from the Tentative list with the lowest cost, move it to the Confirmed list, and return to step 2.

Dijkstra Example

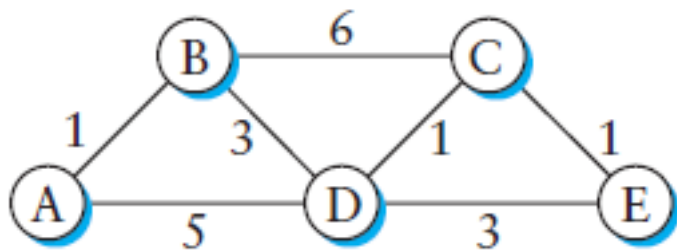


Dijkstra Example



Step	Confirmed	Tentative	Comments
1	(D,0,-)		Since D is the only new member of the confirmed list, look at its LSP.
2	(D,0,-)	(B,11,B) (C,2,C)	D's LSP says we can reach B through B at cost 11, which is better than anything else on either list, so put it on Tentative list; same for C.
3	(D,0,-) (C,2,C)	(B,11,B)	Put lowest-cost member of Tentative (C) onto Confirmed list. Next, examine LSP of newly confirmed member (C).
4	(D,0,-) (C,2,C)	(B,5,C) (A,12,C)	Cost to reach B through C is 5, so replace (B,11,B). C's LSP tells us that we can reach A at cost 12.
5	(D,0,-) (C,2,C) (B,5,C)	(A,12,C)	Move lowest-cost member of Tentative (B) to Confirmed, then look at its LSP.
6	(D,0,-) (C,2,C) (B,5,C)	(A,10,C)	Since we can reach A at cost 5 through B, replace the Tentative entry.
7	(D,0,-) (C,2,C) (B,5,C) (A,10,C)		Move lowest-cost member of Tentative (A) to Confirmed, and we are all done.

Dijkstra Example 2



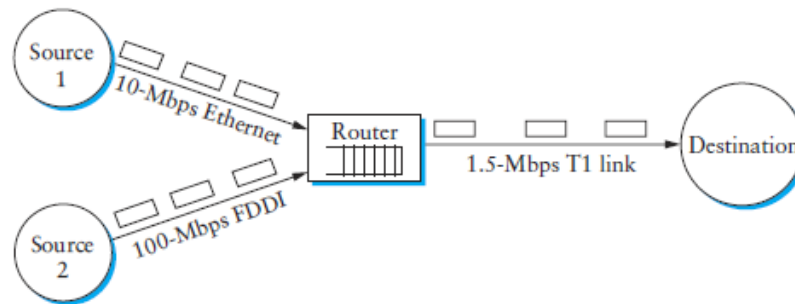
Step	Confirmed	Tentative
1	(A,0,-)	
2	(A,0,-)	(B,1,B) (D,5,D)
3	(A,0,-) (B,1,B)	(D,4,B) (C,7,B)
4	(A,0,-) (B,1,B) (D,4,B)	(C,5,B) (E,7,B)
5	(A,0,-) (B,1,B) (D,4,B) (C,5,B)	(E,6,B)
6	(A,0,-) (B,1,B) (D,4,B) (C,5,B) (E,6,B)	

So Far

- End to End Example
- OSPF
- Congestion Control
 - Queuing

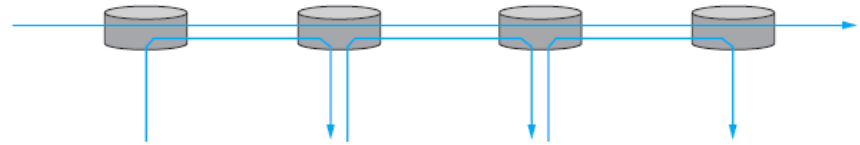
Resource Allocation

- When we have a real network we must deal with contention and congestion
 - Too many users, not enough resources
- We'll talk about packet switched networks for now
- Congestion can come from:
 - Too many users trying to make small connections
 - A few users making huge connections
 - Fast links that must pass over a slower link

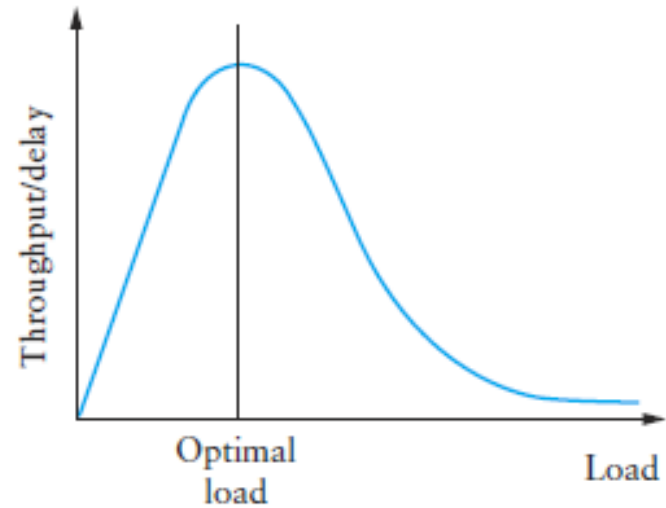


What is the Goal?

- Fairness



- Utilization



What are we Managing?

- Connectionless Flows
 - Data sent between sender and receiver
 - The routers sees them as moving between addresses (ignore ports)
- Routers maintain *soft state* about connections
 - Detected automatically
 - Lives and dies as the connection does
 - Helps the router make better routing decisions
- Flows can be *explicit* or *implicit*
 - Difference is whether the end points tell the routers before they start
 - Datagram versus Virtual Circuits

What is the Network Offering?

- The basic model: Best Effort
 - Try, but no guarantee
 - All packets are created (more or less) equal
- More advanced: Quality of Service (QoS)
 - Senders and receivers *request* the routers to guarantee a minimum amount of resources
 - Some protocols: RSVP, ATM

How are we Managing?

- Router Centric vs. Host Centric
 - Who is doing most of the decision making?
 - Router Centric – the router tells the hosts how fast they can send
 - Host Centric – the hosts decide how fast to send based on their experiences
- Reservation Based vs. Feedback Based
 - Reservation: send request before
 - Requires Router Centric
 - Feedback: change based on what happens
 - Explicit – Router more involved
 - Implicit – Host more involved
- Window Based vs. Rate Based

What is Common?

- With Best Effort:
 - Feedback - since we can't reserve, and therefore...
 - Host centric, and often...
 - Window based
- With QoS:
 - Reservation – normally, and therefore...
 - Router centric, and therefore often...
 - Rate based

So Far

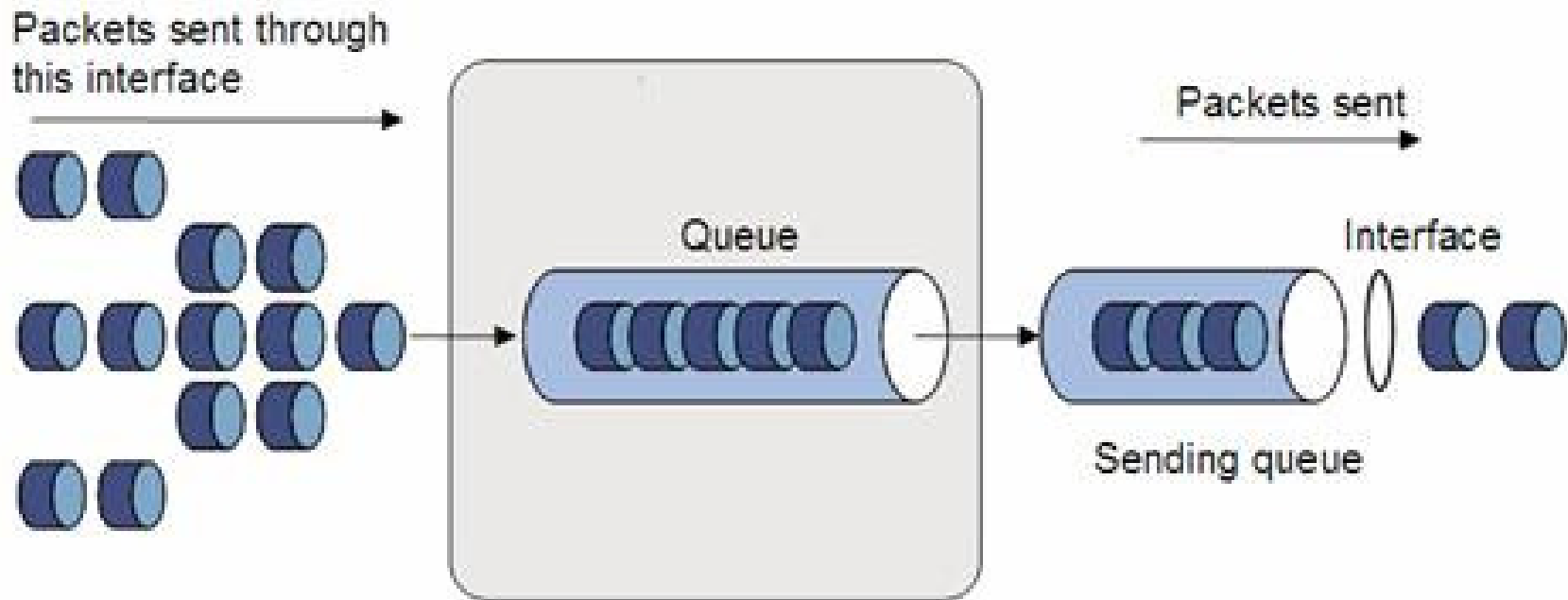
- End to End Example
- OSPF
- Congestion Control
 - Queuing

Queuing Techniques

- First In First Out (FIFO)
- Priority Queuing (PQ)
- Fair Queuing (FQ)
- Weighted Fair Queuing (WFQ)

First In First Out

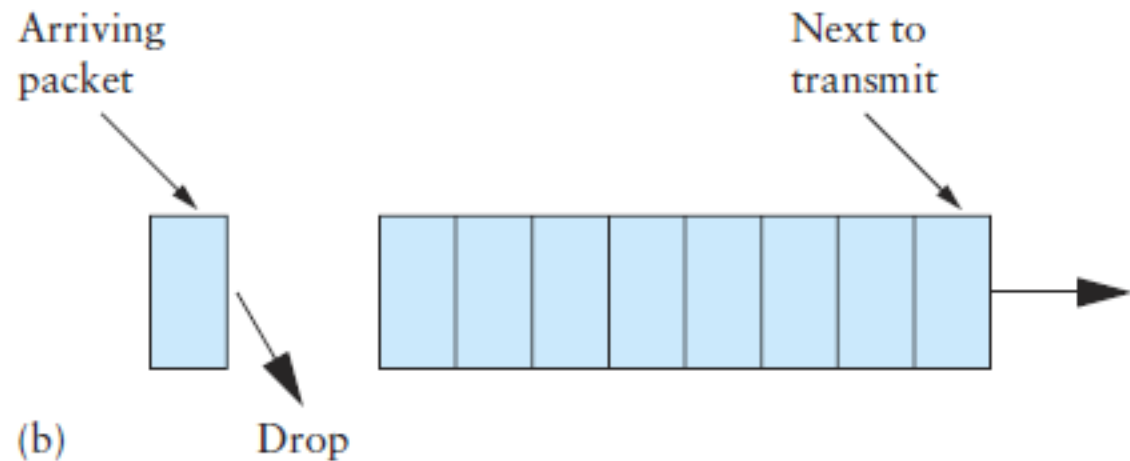
- Rule: Packets are sent out of the router as they arrive



FIFO and Dropping

- What if the queue is full?

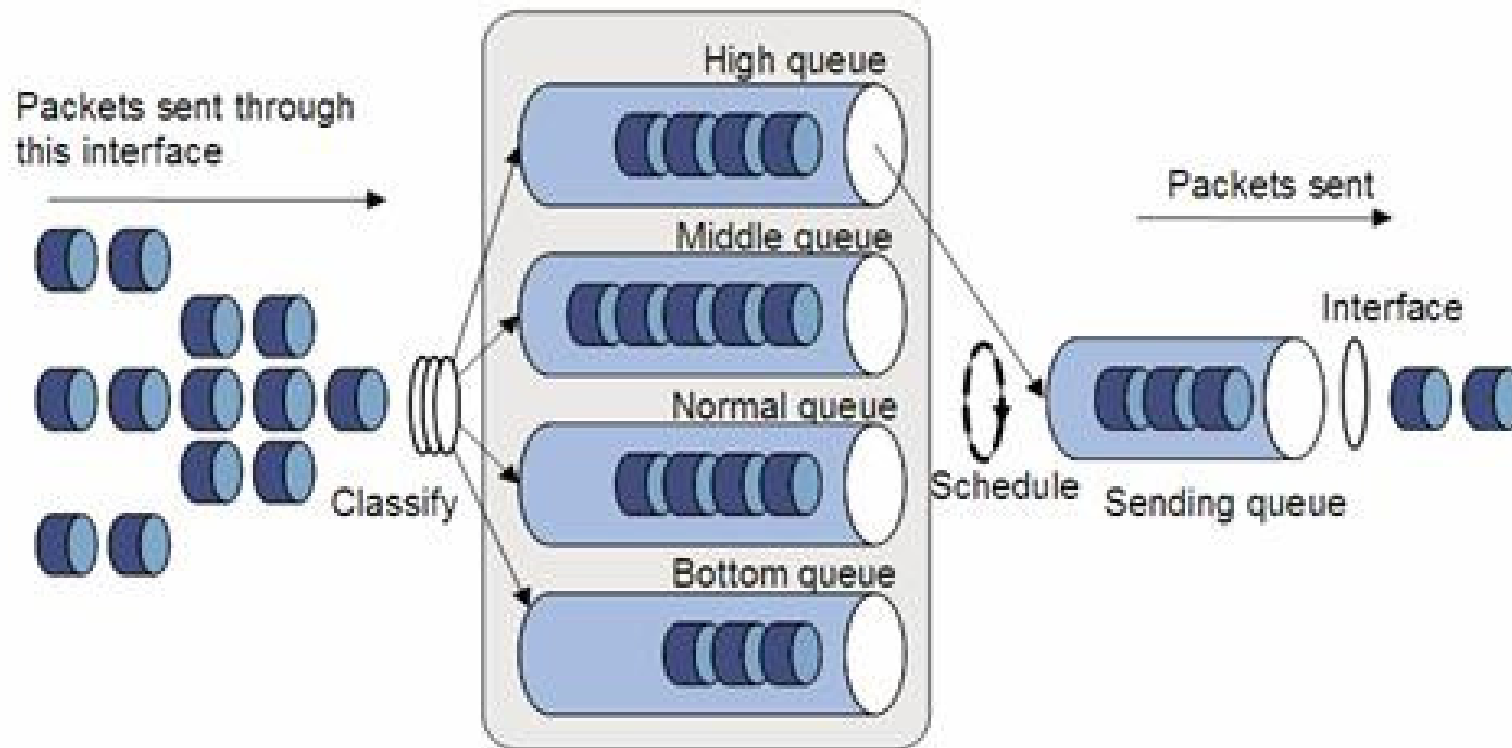
- Drop somebody:
 - Tail Drop



- Random Drop
 - Why?

Priority Queuing

- Put a strict order on the queues
 - Highest priority first, then secondary ones
 - Advantages? Disadvantages?



Fair Queuing

- FIFO can be easily overrun by an out of control sender
 - The router can intervene to make things fairer
- Fair Queuing:
 - Give each flow a queue
 - Manage each flow separately and service them Round Robin
 - If one flow's queue is full, we need to drop (somehow)
 - Each queue gets to send one packet at a time, but we don't interrupt
- But what if one sender sends 1000B packets and another sends 500B packets?
 - To be fair: We want a 1000B packet to “cost” as much as two 500B packets

No Interruptions

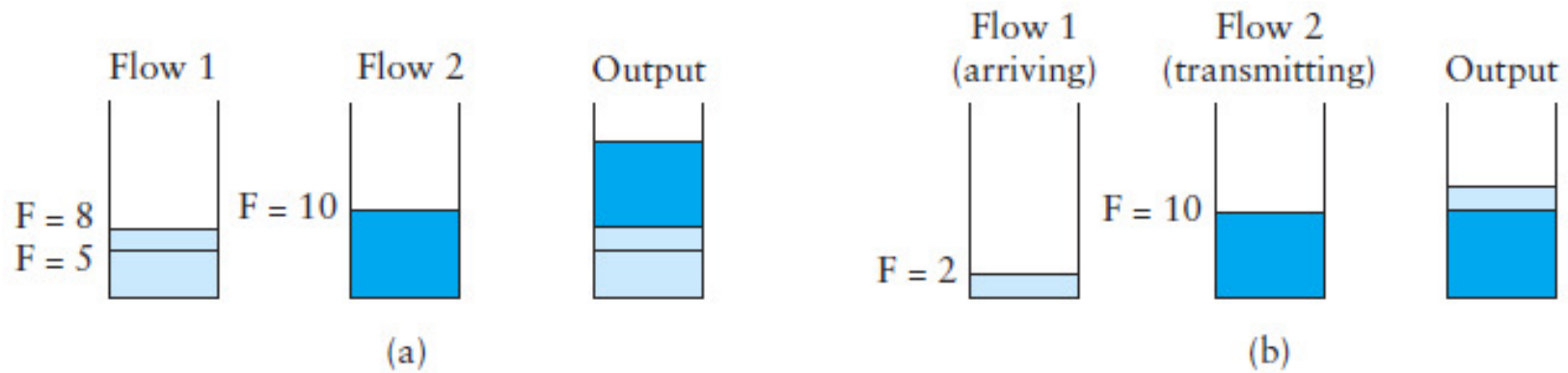


Figure 6.7 Example of fair queuing in action: (a) shorter packets are sent first; (b) sending of longer packet, already in progress, is completed first.

Fair Queuing

- For each packet:
 - Imagine there are no other flows on the router
 - Determine when it *would have finished being sent* based on when it arrived – save it with the packet in the queue
 - Think that a 500b packet takes 500 “ticks” to send
- Packet i arrives at time A_i , it has size P_i , and begins being sent at S_i
 - Then it finishes being sent at $F_i = S_i + P_i$
- What is S_i ?
 - Case 1: There is another packet F_{i-1} from the flow being sent on the line – then $S_i = F_{i-1}$
 - Case 2: The line is free – then $S_i = A_i$
- Result: $F_i = \max(F_{i-1}, A_i) + P_i$

Fair Queuing

- But there are other flows on the router too
 - The “tick” should be when *each flow had a chance to send 1 bit*
 - So slow down the clock calculations for A_i based on the number of active queues
 - If there are 3 active queues, A increments in 0.333 instead of 1
 - This means that the A doesn't match the real “wall clock” time

Conclusion

- End to End Example
- OSPF
- Congestion Control
 - Queuing