Goals of Today's Lecture

- How can we get some sort of reliable performance from the network?
- QoS models and mechanisms:
  - Queuing algorithms beyond FIFO (isolation, fairness)
  - Weighted Fair Queueing
  - Characterizing bursty traffic

Packet Scheduling

- Decide when and what packet to send on output link
  - Usually implemented at output interface of a router

Link Scheduling: FIFO

- What if scheduler uses one first-in first-out queue?
  - Simple to implement
  - But, restrictive in providing guarantees
  - Example: two kinds of traffic
    - Video conferencing needs high bandwidth and low delay
      - E.g., 1 Mbps and 100 msec delay
    - E-mail transfers not very sensitive to delay
    - Cannot admit much e-mail traffic
    - Since it will interfere with the video conference traffic

Link Scheduling: Strict Priority

- Strict priority
  - Multiple levels of priority
  - Always transmit high-priority traffic, when present
  - ... and force the lower priority traffic to wait
- Isolation for the high-priority traffic
  - Almost like it has a dedicated link
  - Except for the (small) delay for packet transmission
    - High-priority packet during transmission of low-priority
    - Router completes sending the low-priority traffic first

Link Scheduling: Weighted Fairness

- Limitations of strict priority
  - Lower priority queues may starve for long periods
  - ... even if the high-priority traffic can afford to wait
  - Traffic still competes inside each priority queue
- Weighted fair scheduling
  - Assign each queue a fraction of the link bandwidth
  - Rotate across the queues on a small time scale
  - Send extra traffic from one queue if others are idle

50% red, 25% blue, 25% green
Max-Min Fairness
- Denote
  - \( C \) – link capacity
  - \( N \) – number of flows
  - \( r_i \) – arrival rate
- Max-min fair rate computation:
  1. compute \( C/N \) (= the remaining fair share)
  2. if there are flows \( i \) such that \( r_i \leq C/N \)
     then update \( C \) and \( N \)
     
     \[ C = C - \sum_{i: r_i \leq C/N} r_i \]  
     \[ N = N - k \] (for \( k \) such flows)
     and go to 1
  3. if not, \( f = C/N \); terminate
- Flows receive at most the fair rate, i.e., \( \min(f, r_i) \)

Fair Rate Computation: Example 1
- If link congested, compute \( f \) such that
  \[ \sum_i \min(r_i, f) = C \]

Fluid Flow System
- 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 System: Example 1
- Red flow has packets backlogged between time 0 and 10
  - Backlogged flow \( \rightarrow \) flow’s queue not empty
- Other flows have packets continuously backlogged
- All packets have the same size
Implementation In Packet System

- 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 System: Example 1

- Select packets in the order they are transmitted (finish) in the fluid flow system

Packet System: Example 2

- Select packets in the order they are transmitted (finish) in the fluid flow system

Implementation Challenge

- Need to compute the finish time of a packet in the fluid flow system...

- ... but the finish time may change as new packets arrive!

- Need to update the finish times of all packets that are in service in the fluid flow system when a new packet arrives
  - But this is very expensive; a high speed router may need to handle hundred of thousands of flows!

Example

- Four flows, each with weight 1

5 Minute Break

Questions Before We Proceed?
Solution: Virtual Time

- Key Observation: while the finish times of packets may change when a new packet arrives, the order in which packets finish doesn’t!
  - Only the order is important for scheduling
- Solution: instead of the packet finish time maintain the number of rounds needed to send the remaining bits of the packet (virtual finishing time)
  - Virtual finishing time doesn’t change when the packet arrives
- System virtual time – index of the round in the bit-by-bit round robin scheme

System Virtual Time: V(t)

- Measure service, instead of time
- V(t) slope – normalized rate at which every backlogged flow receives service in the fluid flow system
  - $C$ – link capacity
  - $N(t)$ – total weight of backlogged flows in fluid flow system at time $t$

$$ \frac{dV(t)}{dt} = \frac{C}{N(t)} $$

System Virtual Time (V(t)): Example 1

- $V(t)$ increases inversely proportionally to the sum of the weights of the backlogged flows

Fair Queueing Implementation

- Define
  - $F_k^i$ - virtual finishing time of packet $k$ of flow $i$
  - $a_k^i$ - arrival time of packet $k$ of flow $i$
  - $L_k^i$ - length of packet $k$ of flow $i$
  - $w_i$ – weight of flow $i$

- The finishing time of packet $k+1$ of flow $i$ is
  $$ F_{k+1}^i = \max( V(a_{k+1}^i), F_k^i ) + L_k^i / w_i $$

Properties of WFQ

- Guarantee that any packet is transmitted within $\frac{\text{packet length}}{\text{link capacity}}$ of its transmission time in the fluid flow system
  - Can be used to provide guaranteed services

- Achieve max-min fair allocation
  - Can be used to protect well-behaved flows against malicious flows
Hierarchical Link Sharing

- Resource contention/sharing at different levels
- Resource management policies should be set at different levels, by different entities
  - Resource owner
  - Service providers
  - Organizations
  - Applications

Hierarchical WFQ (H-WFQ) Example

- Red session has packets backlogged at time 5
- Other sessions have packets continuously backlogged
  
First red packet arrives at 5 ...and it is served at 7.5

Packet Approximation of H-WFQ

- Idea 1
  - Select packet finishing first in H-WFQ assuming there are no future arrivals
  - Problem:
    - Finish order in system dependent on future arrivals
    - Virtual time implementation won't work
- Idea 2
  - Use a hierarchy of WFQ to approximate H-WFQ

Problems with Idea 1

- The order of the 4th blue packet finish time and of the first green packet finish time changes as a result of a red packet arrival
  
Make decision here

Green packet finish first

Problem with Idea 2

- A packet on the second level can miss its deadline (finish time)
  
First level packet schedule

Second level packet schedule

First red packet arrives at 5 ...but it is served at 11!

Solution

- Hierarchical-WFQ with a better implementation of WFQ, called Worst-Case Weighted Fair Queueing (WF2Q)

- Main idea of WF2Q
  - Consider for scheduling only eligible packets
  - Eligible packet at time t: a packet that has started being serviced in the fluid flow system at time t
**Example**

Fluid-Flow System

WFQ (smallest finish time first)

WF2Q (smallest eligible finish time first)

**Hierarchical-WF2Q Example**

- In WF2Q, all packets meet their deadlines modulo time to transmit a packet (at the line speed) at each level.

First level packet schedule

First red packet arrives at 0 and it is served at 7

Second level packet schedule

**How to Specify Bursty Traffic**

- Option #1: Specify the maximum bit rate. Problems?
  - Maximum bit rate may be much higher average
  - Reserving for the worst case is wasteful

- Option #2: Specify the average bit rate. Problems?
  - Average bit rate is not sufficient
  - Network will not be able to carry all of the packets
  - Reserving for average case leads to bad performance

- Option #3: Specify the burstiness of the traffic
  - Specify both the average rate and the burst size
  - Allows the sender to transmit bursty traffic
  - … and the network to reserve the necessary resources

**Characterizing Burstiness: Token Bucket**

- Parameters
  - $r$ – average rate, i.e., rate at which tokens fill the bucket
  - $b$ – bucket depth (limits size of burst)
  - $R$ – maximum link capacity or peak rate

- A bit can be transmitted only when a token is available

**Traffic Enforcement: Example**

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

- $T = 0$ : 1 Kb packet arrives

- $T = 1$ ms: 1 Kb packet transmitted

- $T = 2$ ms: 2 Kb packet transmitted

- $T = 4$ ms: 3 Kb packet arrives

- $T = 8$ ms: 3 Kb packet transmitted

**Source Traffic Characterization: Arrival Curve**

- Arrival curve – maximum amount of bits transmitted during any interval of time $\Delta t$

- Use token bucket to bound arrival curve
Arrival Curve: Example

- Arrival curve — maximum amount of bits transmitted during any interval of time $\Delta t$
- Use token bucket to bound arrival curve

QoS Guarantees: Per-hop Reservation

- End-host: specify
  - arrival rate characterized by token bucket with parameters $(b,r,R)$
  - the maximum tolerable delay $D$, no losses
- Router: allocate bandwidth $r_p$, buffer space $B_a$ such that
  - no packet is dropped
  - no packet experiences a delay larger than $D$

Summary

- Basic mechanism for achieving better-than-best-effort performance: scheduling
  - Multiple queues allow priority service
  - Fair queuing provides isolation between flows
- What do you need to know?
  - System virtual time / finish virtual time
  - WFQ properties and implementation in the fluid flow & packet system
  - Link sharing requirements and challenges
  - Arrival & service curve
  - Token-bucket specification
- What you don’t need to know
  - Details of WF2Q
  - How service curve works