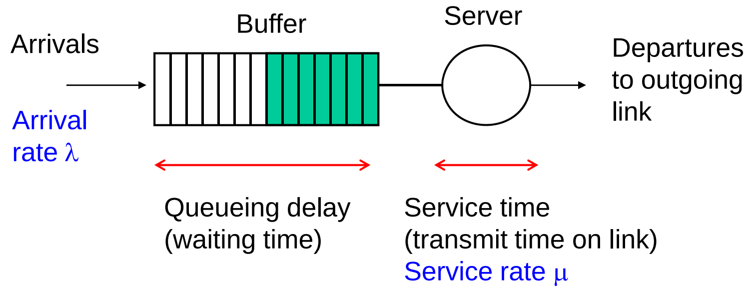
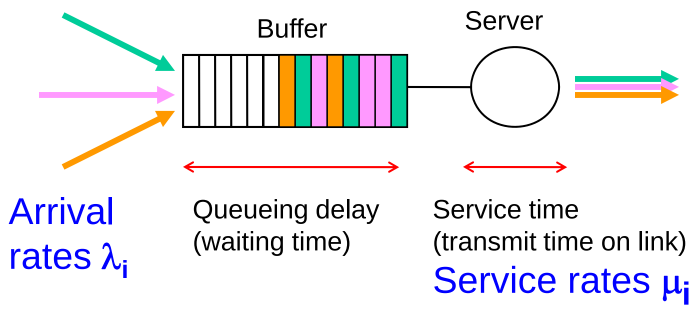
Queuing in a Router or Switch



- **Assume, arrival rate (λ) < service rate (μ).**
  - Needed for stability.
- **Link utilization (ρ) = λ/μ**
  - Mean service time = $1/\mu$.
  - In unit time, λ packets are transmitted, each taking $1/\mu$ time on average.

Multiplexing and Scheduling

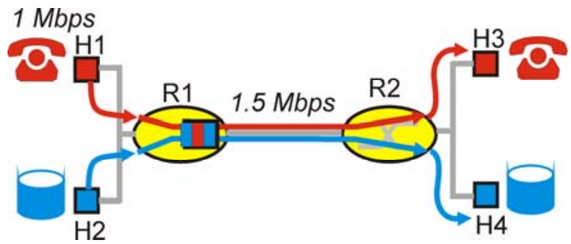


- **A queue can multiplex many "flows" or "connections."**
- **How the server should "schedule" packets from different flows?**
  - Scheduling = mechanism to choose the next packet for transmission. Specifies how the resource (link) should be shared.
  - First-In-First-Out ??

Best Effort and Guaranteed Service

- **Best effort service = No guarantees. Used by adaptive (elastic) applications.**
  - Example: email, file transfer.
- **Guaranteed service = specific service guarantees. Needed by real time applications.**
  - For example, bandwidth, delay, delay jitter, or drop or a combination.
  - Multimedia applications typically will not perform meaningfully if no guarantee
    - Example: Interactive voice needs about 64 kbps BW and 150 ms delay bound.

Where FIFO doesn't work?



- **Think about an IP telephony application and a file transfer with FTP sharing the same links and routers.**
- **FIFO does not differentiate between connections.**
  - Cannot provide guarantees.
  - Cannot prioritize.

Important Conservation Law

- **N connections. Utilization per connection: $\rho_i = \lambda_i / \mu_i$**
- **Mean waiting time for packets belonging to connection $i = q_i$**

$$\sum_{1}^{N} \rho_i q_i = \text{Constant}$$

  - Regardless of scheduling discipline.
  - Assuming, scheduler is work-conserving, i.e., link not idle if packets waiting.
- **Mean waiting time for a connection can be reduced only at the expense of another.**

Scheduling Best Effort Connections with Elastic Traffic

- **Desirable property: "Fairness"**
  - Not an important property for "guaranteed service".
- **The fairness question**
  - Resource capacity C (e.g., link bandwidth).
  - N connections or flows sharing the resource.
  - Each connection has a demand.
  - Sum of individual demands from connections exceeds C. Otherwise, trivial problem.
  - How to fairly allocate resource?

Fairness

- **Could be framed as a more general economic question.**
- **Many definitions possible. Need to form an appropriate definition relevant to context.**
  - Must be able to implement with reasonable efficiency.

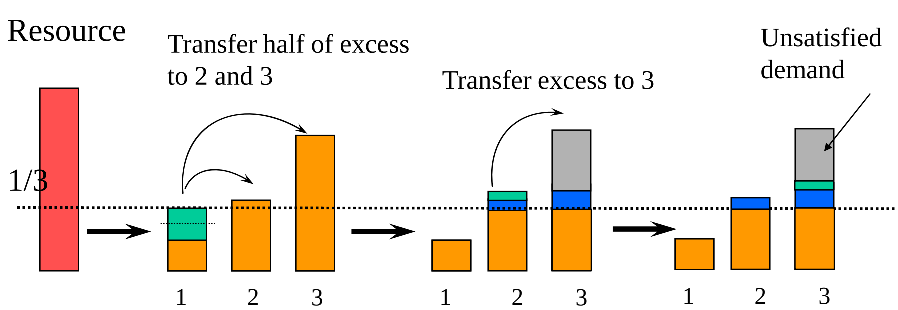
Notion of Max-Min Fairness

- **Basic idea: "Envy-free" allocation. Maximize the minimum.**
  - No connection gets more than what it wants.
  - After a max-min fair allocation, only way to make a connection "richer" will be to make another equal or poorer flow further poorer.
- Formally, assume a feasible allocation $X = (x_1, \ldots, x_n)$ means that $\sum_i^n x_i \leq C$ and $x_i \leq d_i$. If $X = (x_1, \ldots, x_n)$ is the max-min fair allocation, then it is feasible and the following must be true for any alternative feasible allocation $Y = (y_1, \ldots, y_n)$. If $y_i > x_i$, then there must exist some j such that $x_j \leq x_i$ and $y_j < x_j$.

Implementing Max-Min Fairness

- **Intuitive mechanism:**
  - Each connection gets no more than what it wants.
  - If any connection's demand is unmet, then all connections with unmet demand get an equal share.
- **Allocation algorithm:**
  - Sort N connections with increasing demand.
  - Allocate resource equally (C/N).
  - For connection i = 1 to N, do
    - If demand of connection i is less than its current allocation,
    - distribute the excess equally among the rest.

Max-Min Fair Share Allocation Example



Other Notions of Fairness: Proportional Fairness

- Usually used in the context of congestion control.
- Based on a notion of utility function.
- Maximize $\sum_1^N U_i(x_i)$, where x_i is the allocation for connection i, and $U_i()$ is an utility function, modeling i-th connection's utility when it gets an allocation x_i.
- For proportional fairness, the utility function is $U_i(x_i) = \log(x_i)$.

Other Notions of Fairness: Proportional Fairness (contd.)

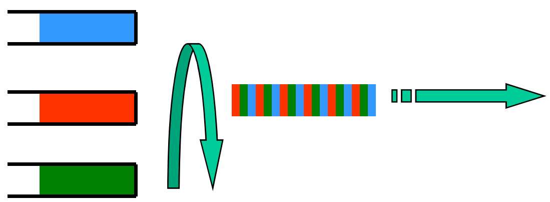
It can shown that for proportional fairness if $X^* = (x_1^*, \ldots, x_n^*)$ is the optimal (i.e., proportionally fair) allocation, then for any other allocation X different from X^*,

$$\sum_1^n \frac{x_i - x_i^*}{x_i^*} \leq 0.$$

- **It can be shown that max-min fairness can also be modeled as utility maximization, however, for a very different definition of utility.**
- **TCP's congestion control also can be shown to a utility maximization for another definition of utility.**

Back to Max-Min Fairness: Scheduling Best Effort Connections

- **Goal: Guarantee max-min fair share of link bandwidth.**
- **Generalized Process Sharing (GPS) Policy**
  - Put each connection to different queues.
  - Visit each non-empty queue in turn.
  - Serve an infinitesimal amount. Note infinitesimal is un-implementable.



Why is GPS Max-min Fair?

- **N connections. Link capacity C.**
- **If all queues non-empty, each connection gets C/N.**
- **If some queues are empty in a turn, they have less than C/N demand**
  - Empty queues are not visited in a turn.
  - Thus, non-empty queues get more "service."
  - Equivalent to sharing the excess.

Max-Min Weighted Fair Share Allocation

- **Assign weights (w_1, w_2, .., w_n) for different sources to indicate their relative share.**
- **Allocate resource in the order of increasing demand, but in proportion to weights.**
- **No connection gets more than its demand.**
- **Excess redistributed in proportion to the weights of the connections with unsatisfied demand.**

Example

Demands	4	2	10	4
Weights	2.5	4	0.5	1
Capacity	16			
Normalize weights	5	8	1	2

- **5+8+1+2 = 16 total shares. Allocate shares to connections in proportion to weights.**

Allocation	5	8	1	2

- **7 shares excess (connections 1 & 2). Redistribute to 3 & 4 in proportion to weights.**

Allocation	4	2	3.33	6.67

- **Redistribute excess at connection 4.**

Allocation	4	2	6	4

Bit-by-Bit Round Robin

- **Similar to GPS.**
- **But still is unrealistic. Need a header for every bit.**
- **Need more realistic solutions.**

Round Robin (RR) and Its Weighted Version (WRR)

- **Serves one packet from each non-empty connection in a round.**
  - Proportional no. for weighted.
- **Simplest emulation of GPS.**
- **If unequal packet sizes (but fixed for a connection), divide weights by packet sizes to get a new set of weights.**
- **If packet size varies, use the mean value.**
  - Problem: May not know the mean ahead of time.

Fairness of WRR

- **Not fair in timescales shorter than a round time.**
  - Of course, in time scales shorter than a packet time any packet-oriented discipline is unfair.
- **Note that the round time could be large for many connections.**

Deficit Round Robin (DRR)

- **Handles variable packet sizes without knowing the mean in advance.**
- **Init: deficit counter = 0.**
- **Visit each non-empty connection and serve a quantum worth of bits in each round.**
- **deficit counter = deficit counter + quantum.**
- **If deficit counter is equal to or larger than the packet at the head of the connection queue**
  - transmit that packet and
  - reduce deficit counter by packet size.
- **For efficiency, DRR should serve at least one packet in each round.**
  - Thus, quantum size = max. possible packet size.

Fairness Measure (FM)

Define,

$$FM(t_1, t_2) = \frac{\text{sent}_i(t_1, t_2)}{f_i} - \frac{\text{sent}_j(t_1, t_2)}{f_j},$$

where $\text{sent}_i(t_1, t_2)$ is the number of bits sent for connection i over the output link in the time interval (t_1, t_2), and f_i is the number of bits would have been sent with a GPS scheduler.

FM is the maximum value of $FM(t_1, t_2)$ over all possible execution of the algorithm and all possible values (t_1, t_2).

Analysis for RR

- **FM for RR algorithm is infinity.**
  - Assume, simple RR. No weights. Packet sizes unknown.
  - In one round the difference can be (Max – Min) packet size.
  - For arbitrarily long intervals, the difference grows arbitrarily.
  - Similar case for intervals smaller than a round even when packet size is known.
- **How about DRR?**

Analysis for DRR

- During any interval (t_1,t_2), if connection i is served m times, another connection j will be served (m-1) times at the minimum.
- If a connection i is served m times, the following must be true.

$$mQ - \text{Max} \leq \text{sent}_i(t_1, t_2) \leq mQ + \text{Max},$$

where Q is the quantum size and Max is the max packet size.

Analysis of DRR (contd)

- Thus, in the worst case, connection i can get mQ + Max, and connection j can get (m-1)Q – Max.
- So, the difference = 2Max + Q.
- If Q = Max (as suggested), FM = 3Max.
- Note again that any packet-oriented discipline will not be better than FM = Max.

Thought Question

- **Suppose, we make the quantum size = 1 bit.**
- **Is DRR same as bit-by-bit round robin?**
- **Prove or disprove.**

Fair Queuing (FQ)

- **Intuition: Emulate GPS on the side. Transmit packets in the order in which a GPS server would serve the end of the packet.**
  - That is, transmit the packets in the order of finishing times of the packets.
- **Round no. = #rounds completed by a bit-by-bit RR server**
  - Could be fractional.
  - 3.5 means 3 rounds completed, halfway through 4th.
  - *Since some connections may be inactive all rounds are not of same length.*

FQ Contd.

- **Finish no. of a packet = round no. at which a bit-by-bit RR server would complete serving the whole packet.**
- **Example:**
  - Packet (size 10 bits) arrives at round no. 3. This is the head packet for this connection. Then, finish no. = 10+3 = 13.
  - If packet arrives in an active connection, finish no. = finish no. of last packet for that connection (>3) + packet size in bits.

FQ Contd.

- **P(i,k,t) = Size of k-th packet arriving on connection i at time t.**
- **F(i,k-1,t) = Finish no. of (k-1)th packet for that connection.**
- **R(t) = round no. at time t.**
- **F(i,k,t) = max {F(i,k-1,t),R(t)} + P(i,k,t)**
- **Note hard part is to know R(t) without actually simulating a bit-by-bit RR server.**

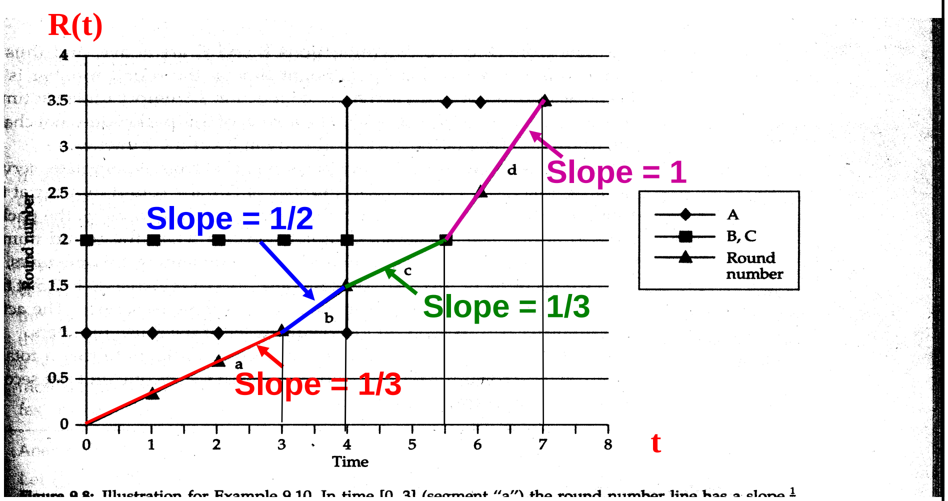
Round Number in FQ

- **Round number does not increase with a constant rate with time.**
  - Depends on no. of active connections.
  - $\delta R(t)/dt = C / N_{conn}$, where C is the link speed (capacity) and N_{conn} is the # active connections.
  - It is more appropriate to view R(t) as each active connection's instantaneous share of link bandwidth.
  - Thus, R(t) is a real no. that increases at a rate inversely proportional to N_{conn}
  - With this view FQ emulates GPS rather than bit-by-bit RR.

Finish Number in FQ

- **Finish no. is independent of no. of active connections (current or future), future packet arrivals etc.**
  - Note again finish no. not same as finish time.

Example from Keshav's Book



Designing FQ Scheduler

- **As soon as a packet arrives at time t, scheduler computes R(t).**
  - Use last known value of N_{conn} to compute slope.
  - N_{conn} may be overestimated as some connections may turn inactive by this time.Thus R(t) will be underestimated.
  - **Iterated deletion technique**: If some connection's finish time < R(t), re-compute R(t), until no such inactive connections.
- **Calculate finish no.**
  - If arriving on active connection
  - finish no. = previous finish no. + size
  - Else, finish no. = R(t) + size.

Designing FQ Scheduler (contd.)

- **Put packets in a priority queue ordered by finish no.**
  - If no space, drop packet(s) with largest finish number(s). This is also max-min fair buffer allocation.
- **On end of transmission, select next packet with the lowest finish no.**
- **Complexities:**
  - Priority queue: O(log n) insert; O(1) delete.
  - Computing round no. on packet arrival.
  - Maintaining per connection state (e.g., finish no.)
  - Doing all these at link speed (~100Mb/s-1Gb/s) !!

Weighted Fair Queueing

- **Similar to FQ. Just add weights.**
- **$F(i,k,t) = \max\{F(i,k-1,t), R(t)\} + P(i,k,t)/\phi_i(t)$.**
  - $\phi_i(t)$ = weight of the i-th connection.
  - R(t) vs. t slope = 1/ sum of weights of all active connections.

How Good is Fair Queuing?

- Fairness measure FM = Max.
- This is because after a connection sends out a max sized packet, it cannot send another immediately after if another connection is also continuously backlogged.

	Fairness Measure (FM)	Computation per packet
Fair Queuing	Max	O(log n)
Deficit Round Robin	3Max	O(1)