Weighted Fair Queuing

"A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks",

A. Parekh, R. Gallager, IEEE/ACM Trans. on Networking June 1993, April 1994

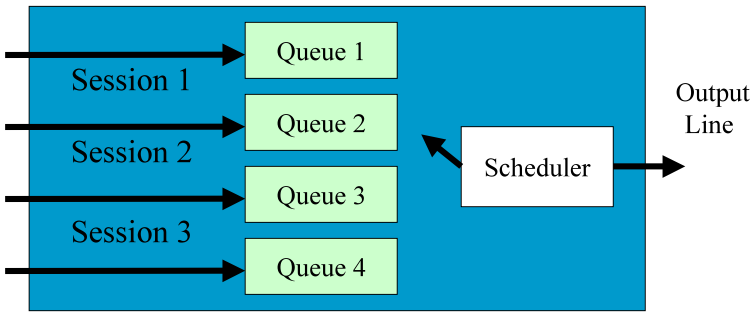
"WF^{2}Q: Worst Case Fair Weighted Fair Queuing",

Jon Bennett, Hui Zhang, IEEE INFOCOM 1996

Generic Switch Structure

- A Scheduling Discipline *resolves contention*. Decides who's next?
- Can differentiate users / classes of traffic.

Switch Output Module

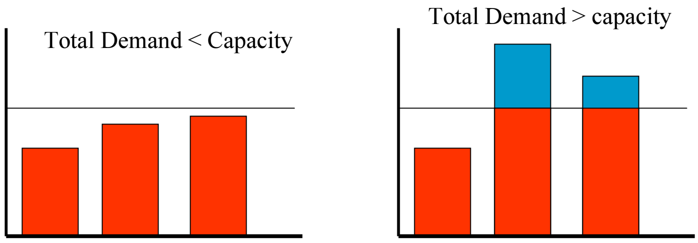


Scheduling

- Desired Properties
  - *Fair* resource sharing.
  - Performance guarantees.
  - Offer different users different quality of services
- Where?
  - Wherever contention for resources occurs.
  - We will concentrate on the output of a network layer switch.

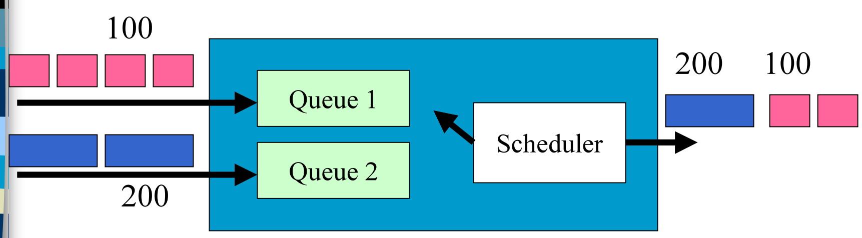
Max-Min Fairness

- Assume all users have equal *rights* (priority) to a resource:
  - Each connection gets no more than what it wants.
  - Excess is distributed evenly across connections.
- Possible to generalize for the *weighted* case: Some connections are more important than others.



Fairness

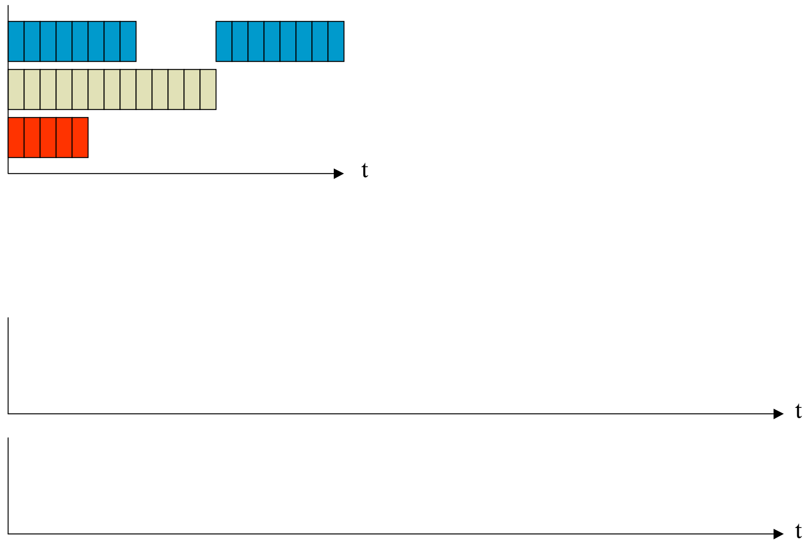
- Fairness is *intuitively* a good idea.
- But it also provides *protection*
  - traffic hogs cannot overrun others
  - No need to be aggressive to get what you want.



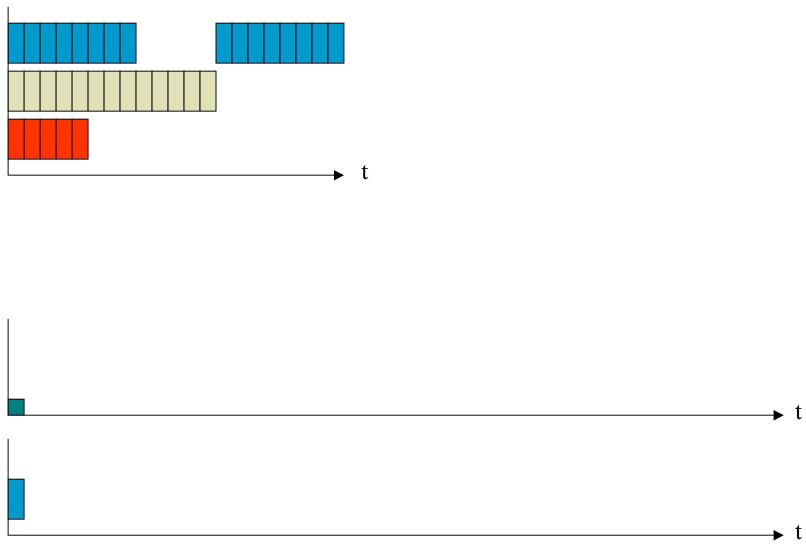
Outline

- Introduction to Link Scheduling
- Generalized Processor Sharing (GPS)
- Weighted Fair Queuing
- Worst Case Fair WFQ (WF^{2}Q)
- GPS Performance Guarantees
- Real World Use

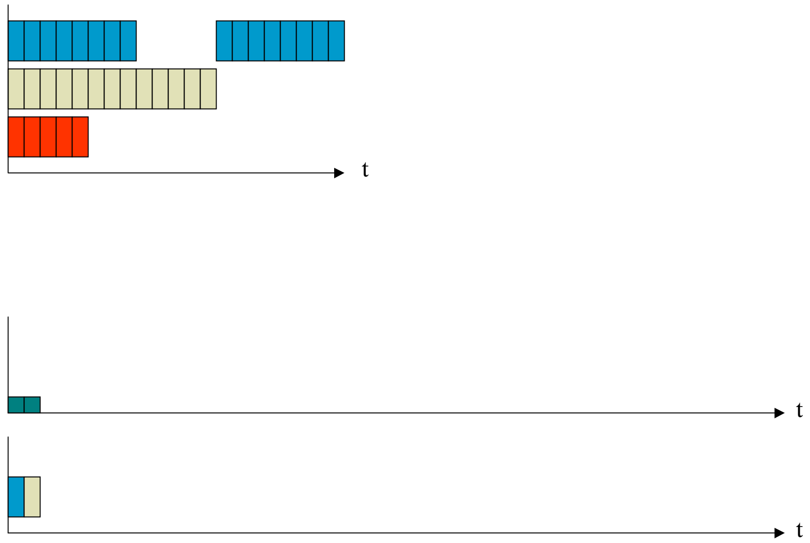
Bit Round Robin Scheduling



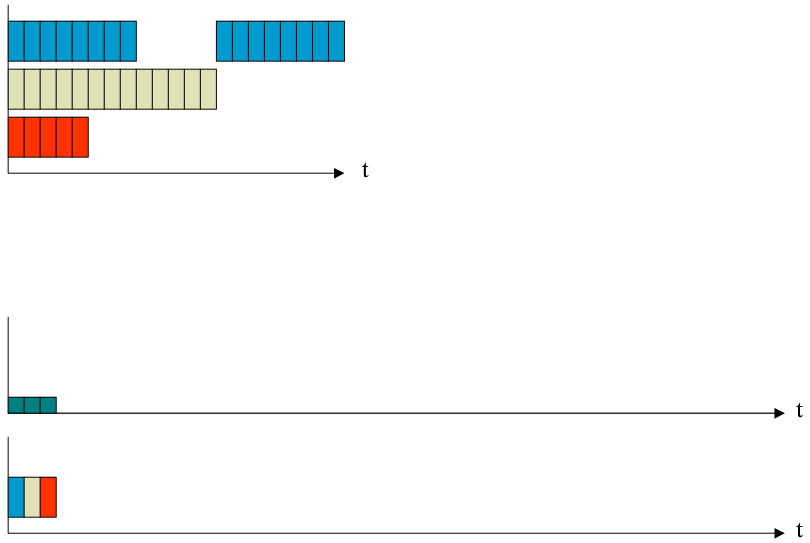
Bit Round Robin Scheduling



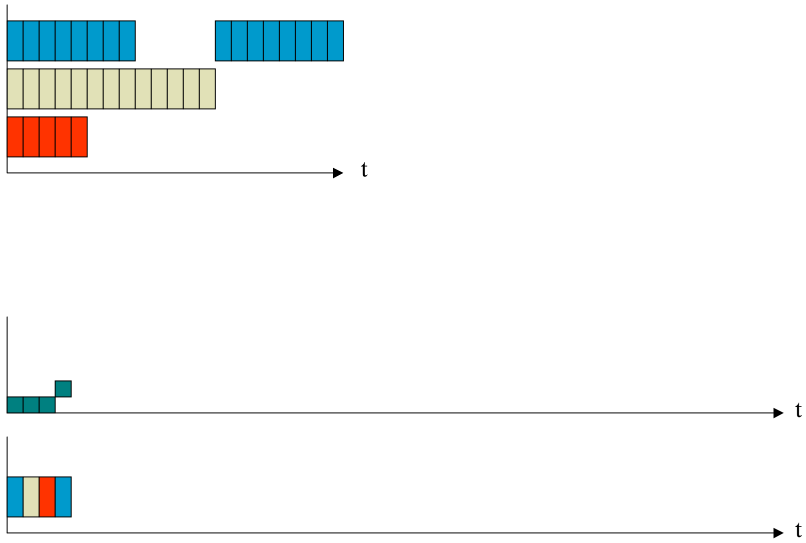
Bit Round Robin Scheduling



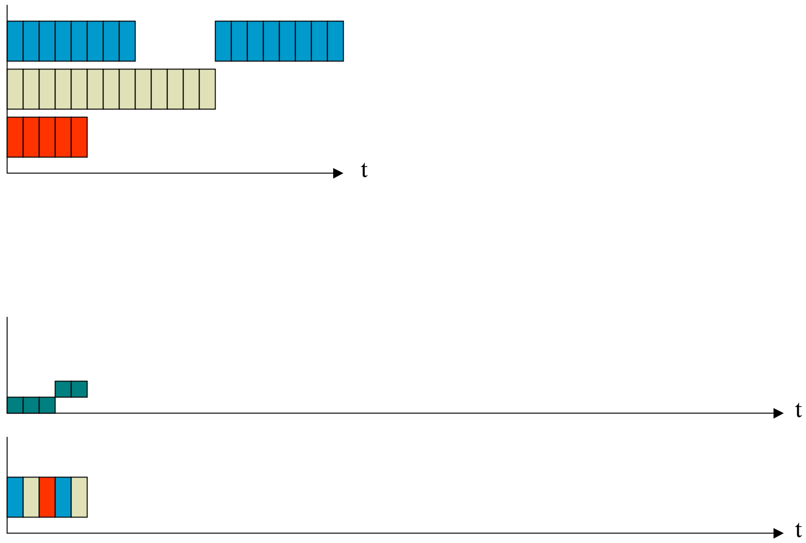
Bit Round Robin Scheduling



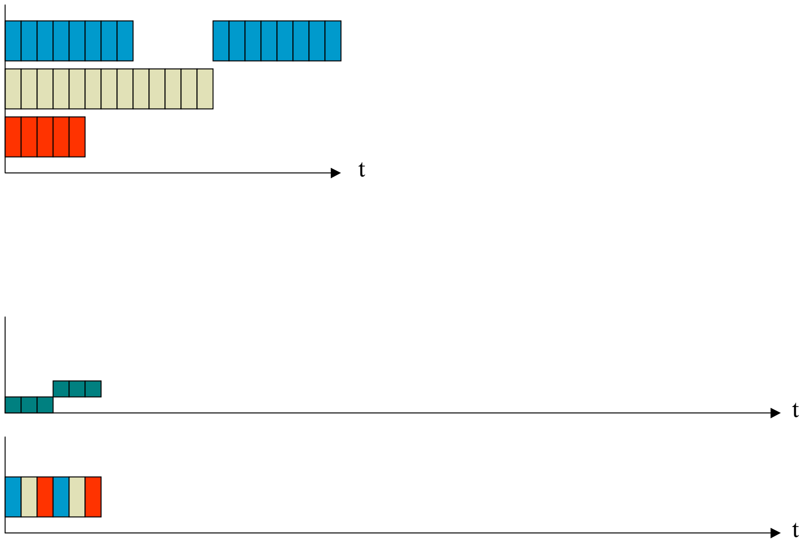
Bit Round Robin Scheduling



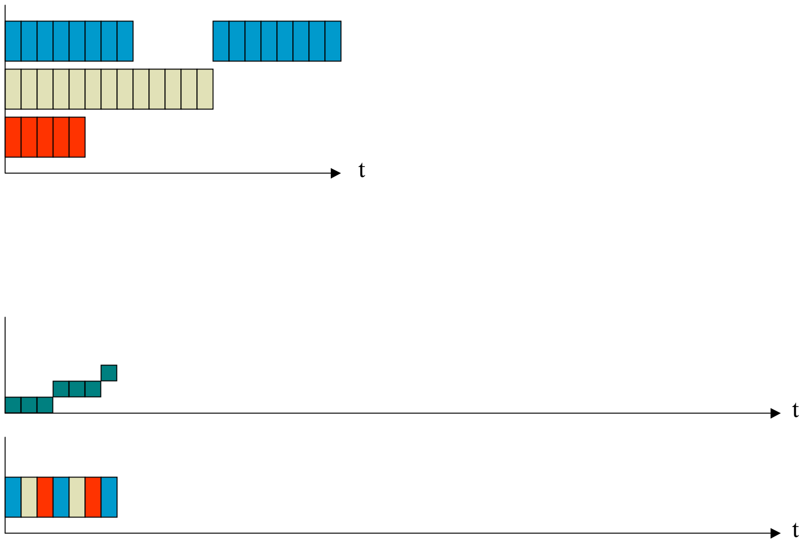
Bit Round Robin Scheduling



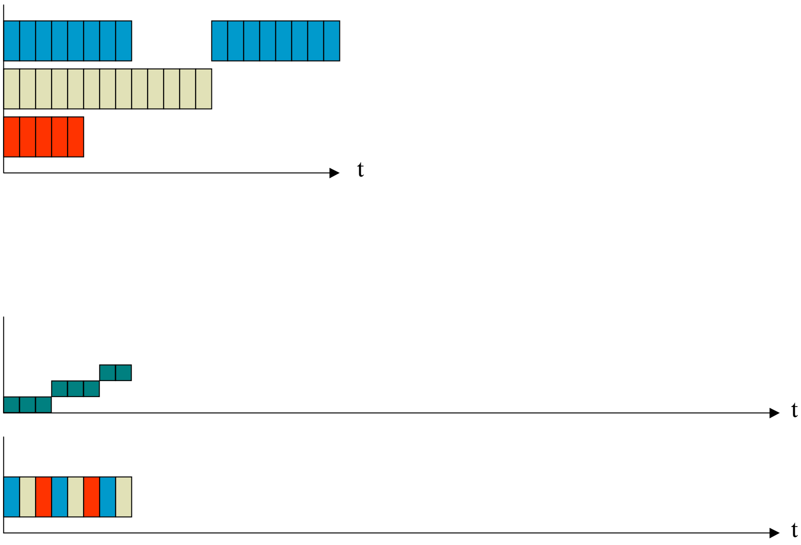
Bit Round Robin Scheduling



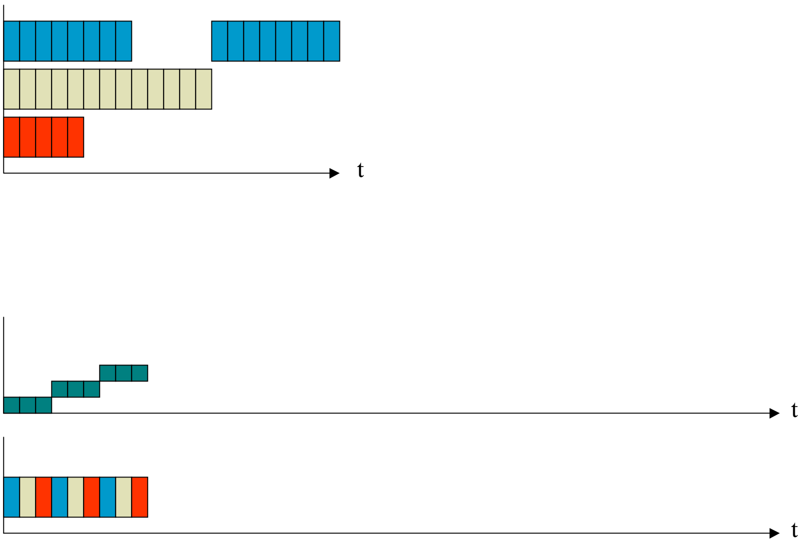
Bit Round Robin Scheduling



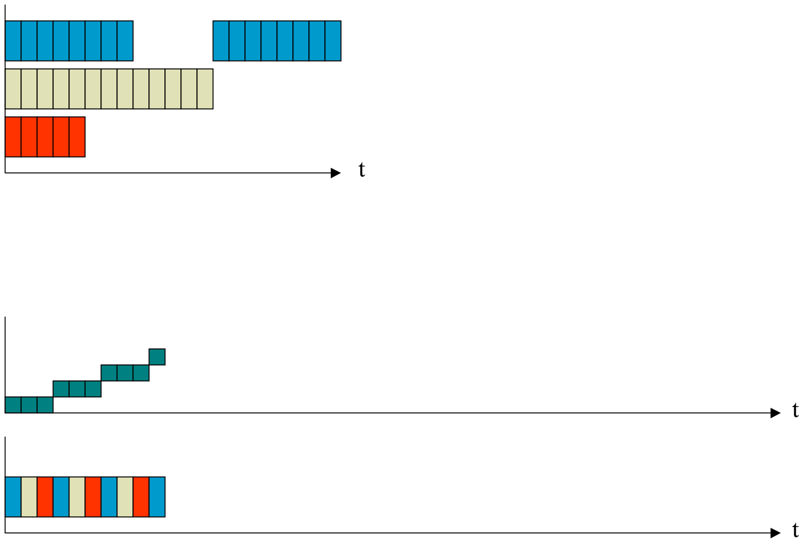
Bit Round Robin Scheduling



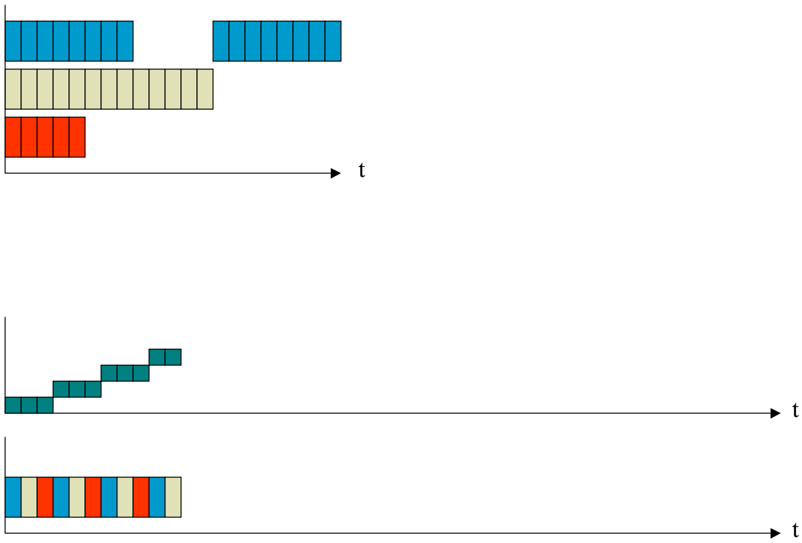
Bit Round Robin Scheduling



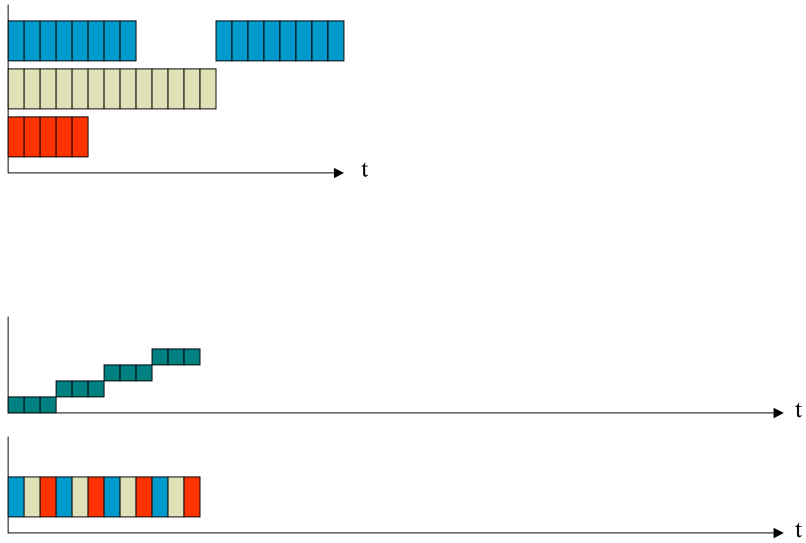
Bit Round Robin Scheduling



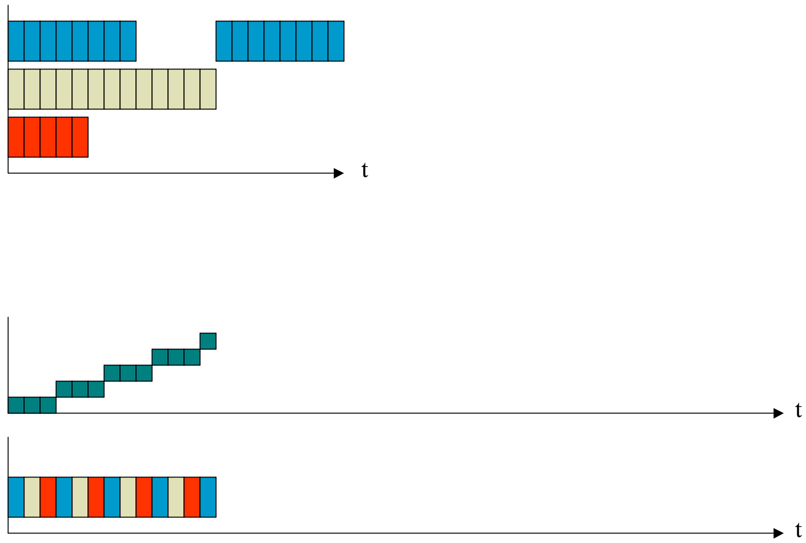
Bit Round Robin Scheduling



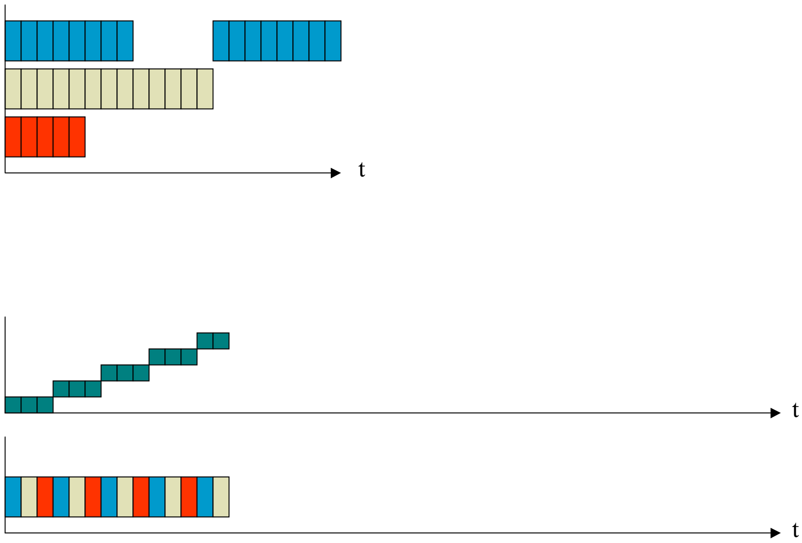
Bit Round Robin Scheduling



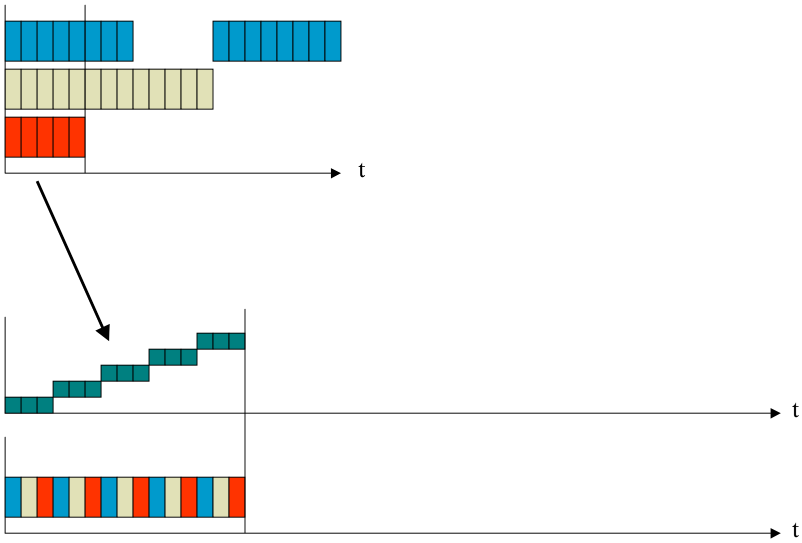
Bit Round Robin Scheduling



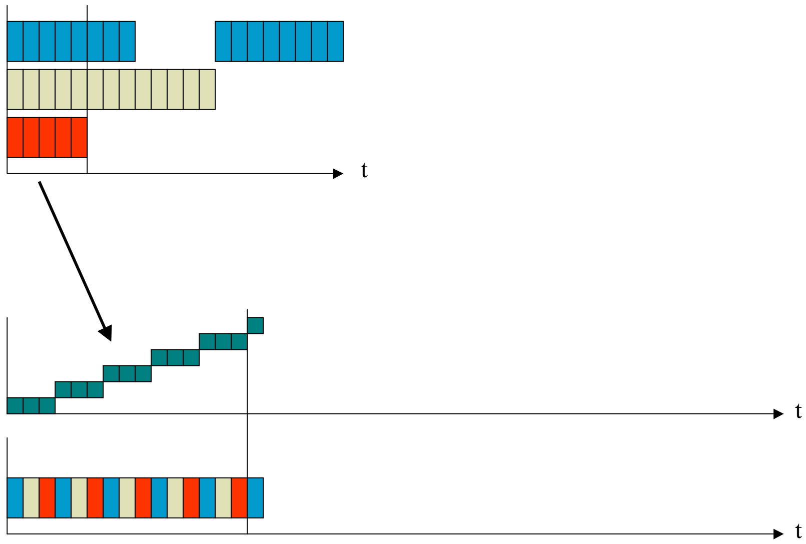
Bit Round Robin Scheduling



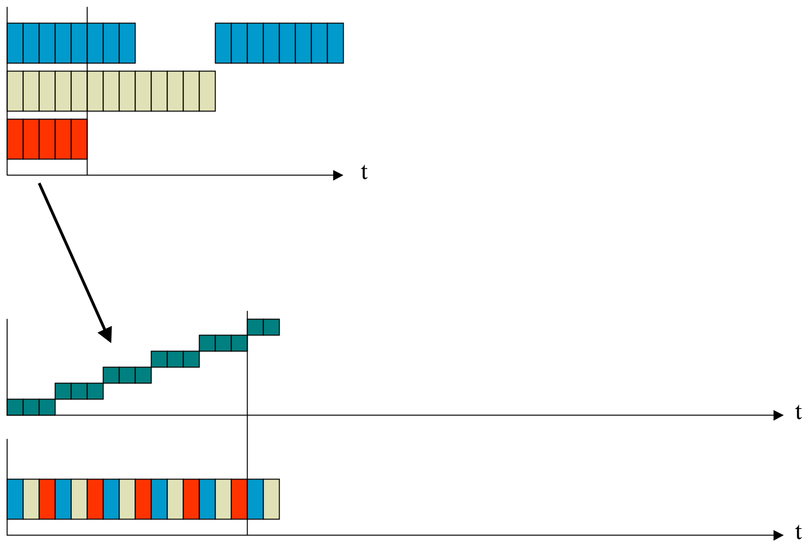
Bit Round Robin Scheduling



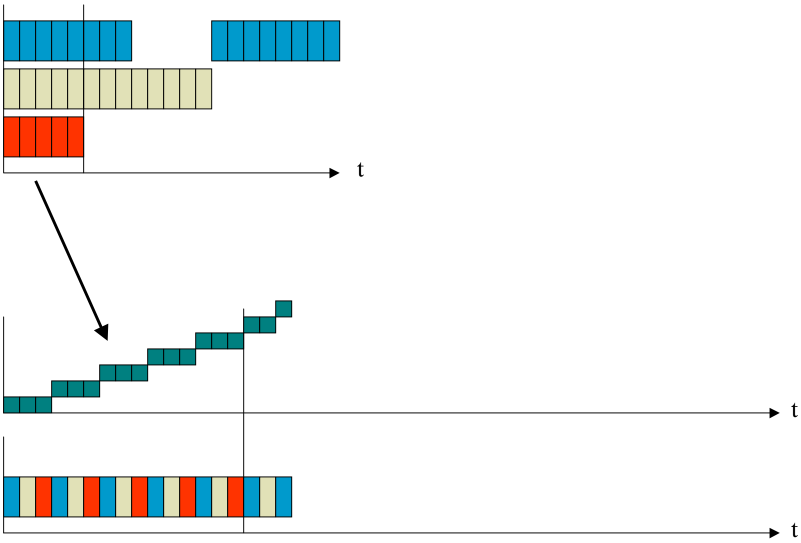
Bit Round Robin Scheduling



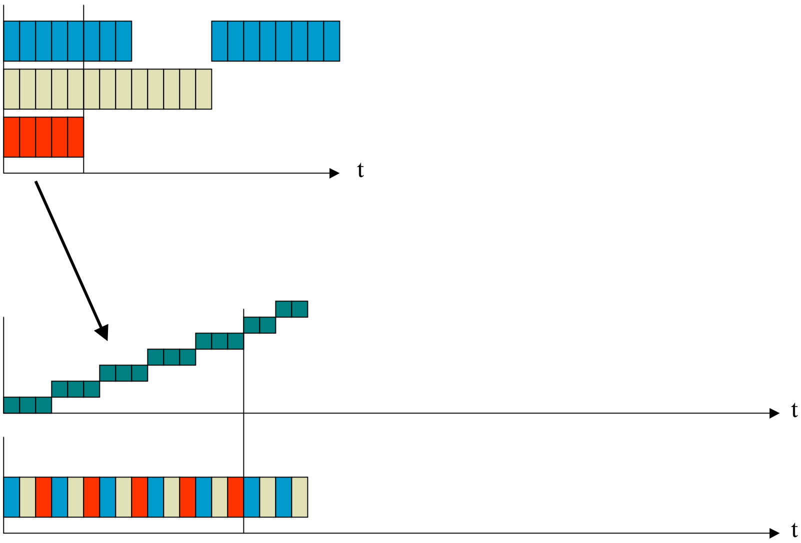
Bit Round Robin Scheduling



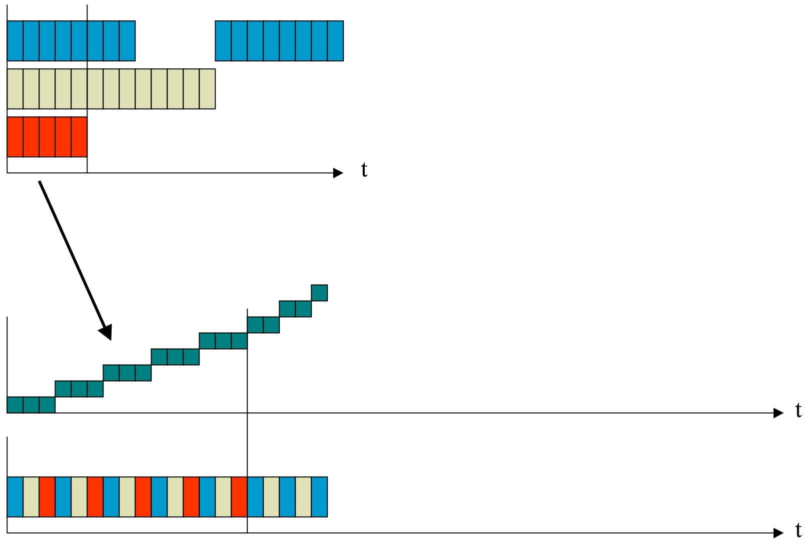
Bit Round Robin Scheduling



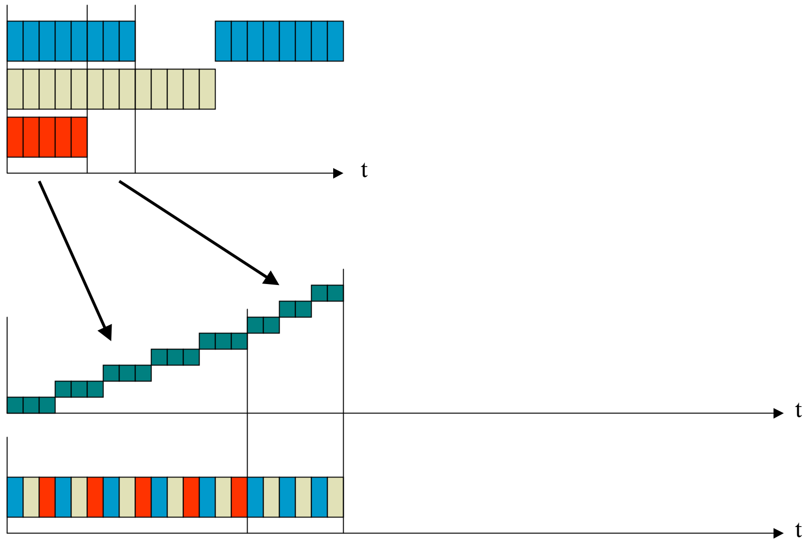
Bit Round Robin Scheduling



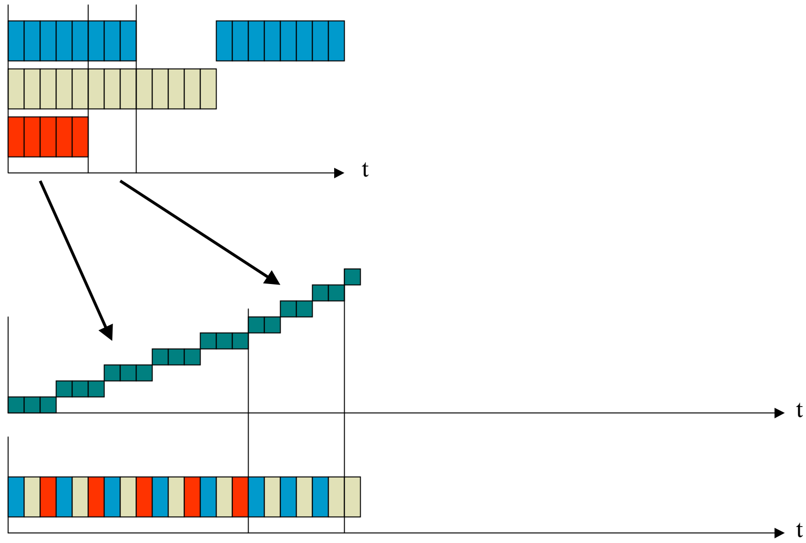
Bit Round Robin Scheduling



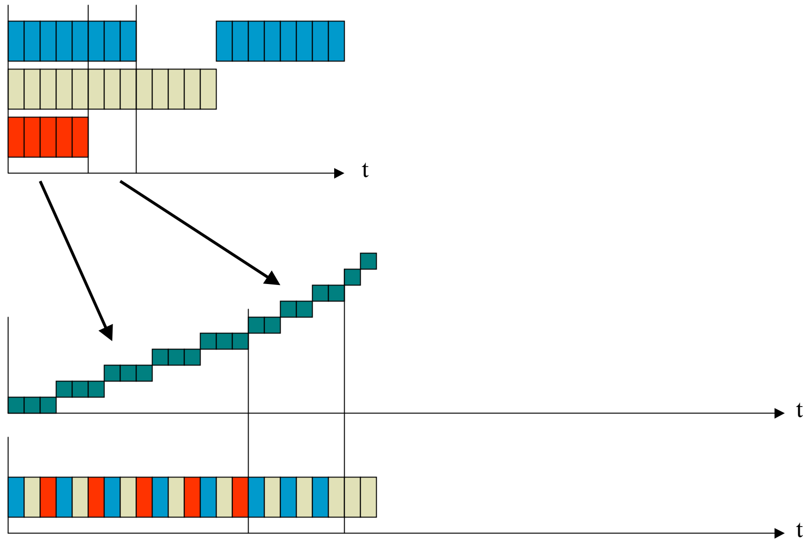
Bit Round Robin Scheduling



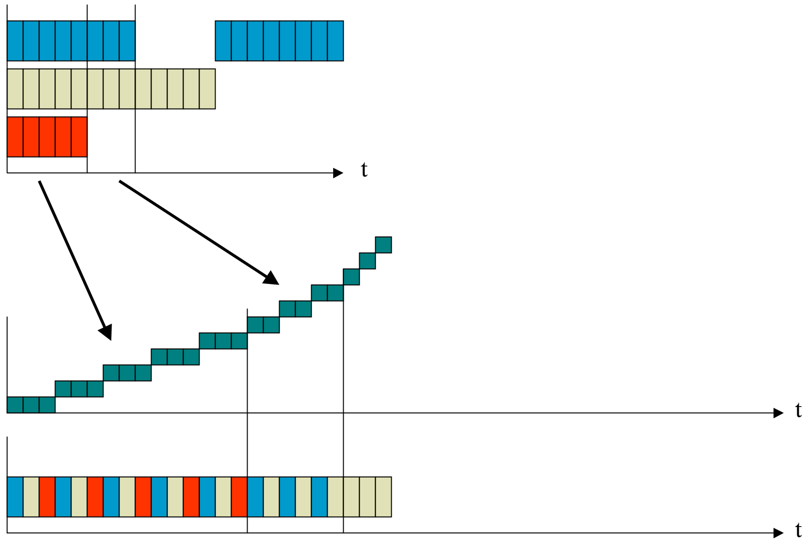
Bit Round Robin Scheduling



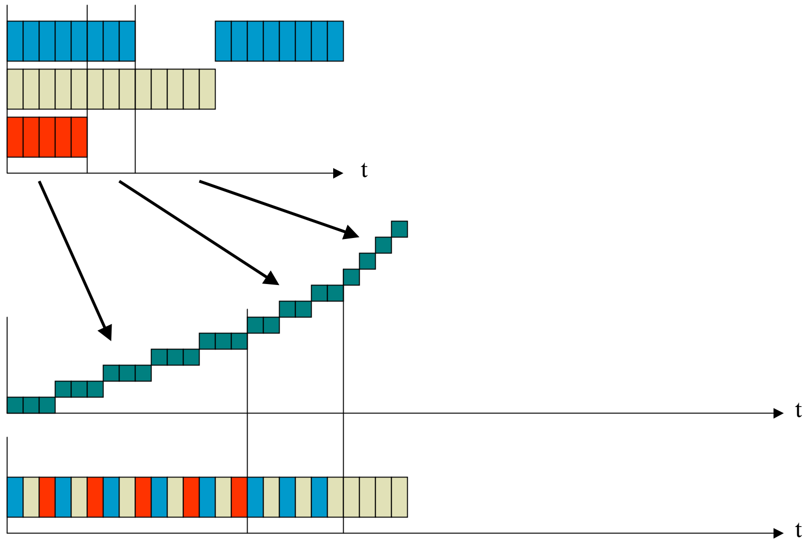
Bit Round Robin Scheduling



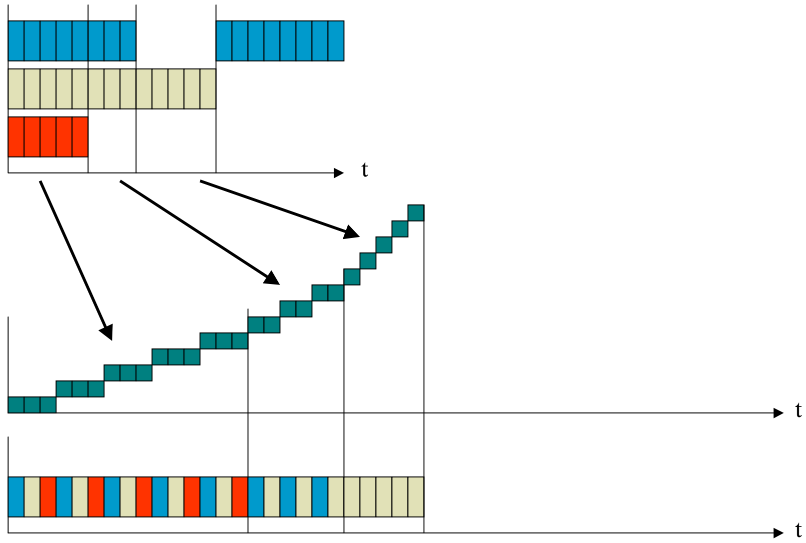
Bit Round Robin Scheduling



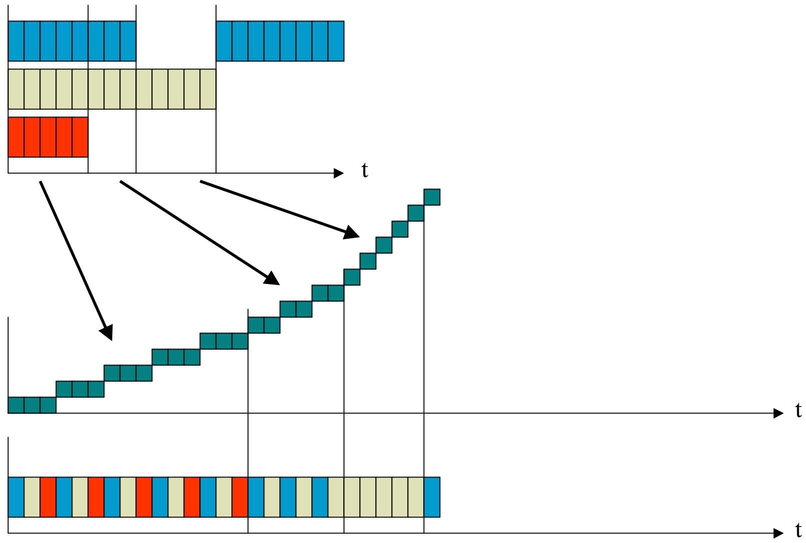
Bit Round Robin Scheduling



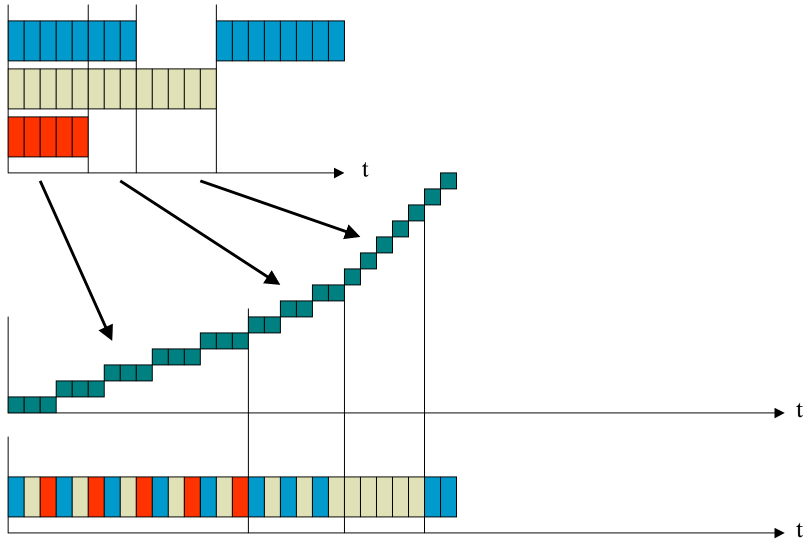
Bit Round Robin Scheduling



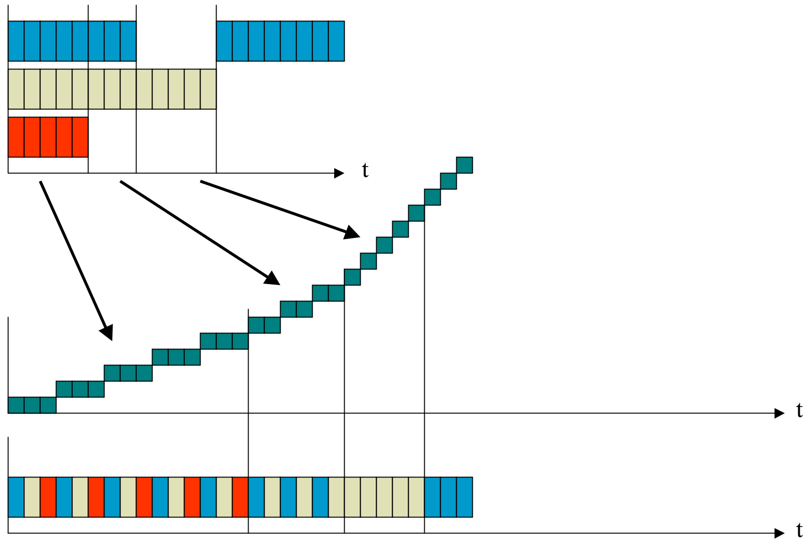
Bit Round Robin Scheduling



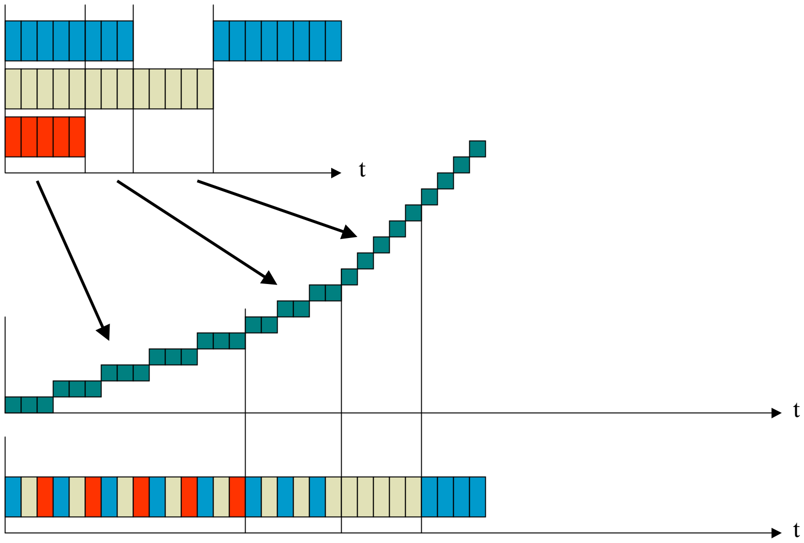
Bit Round Robin Scheduling



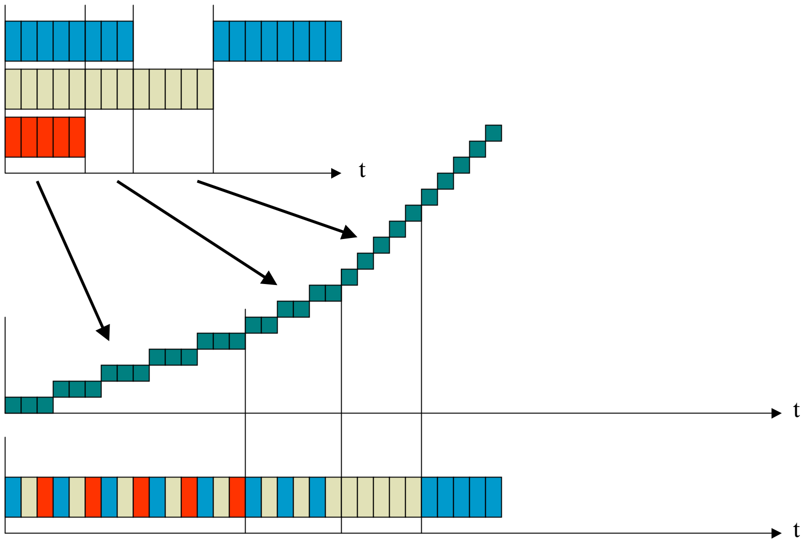
Bit Round Robin Scheduling



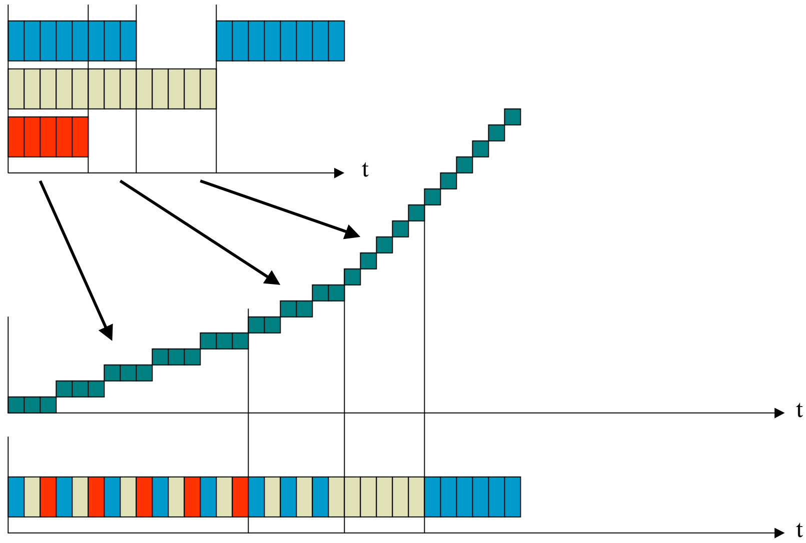
Bit Round Robin Scheduling



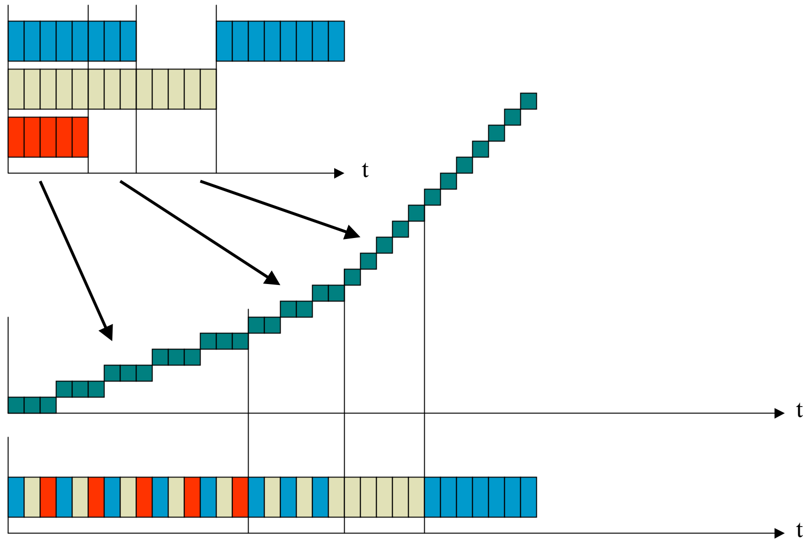
Bit Round Robin Scheduling



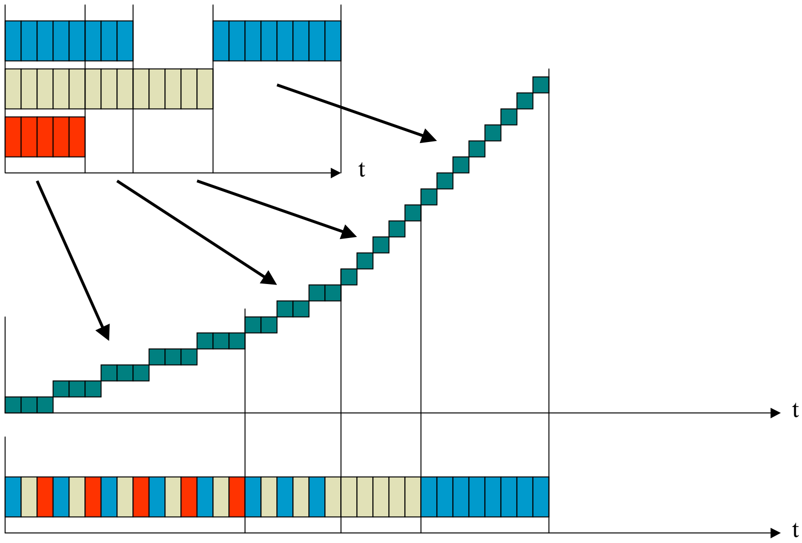
Bit Round Robin Scheduling



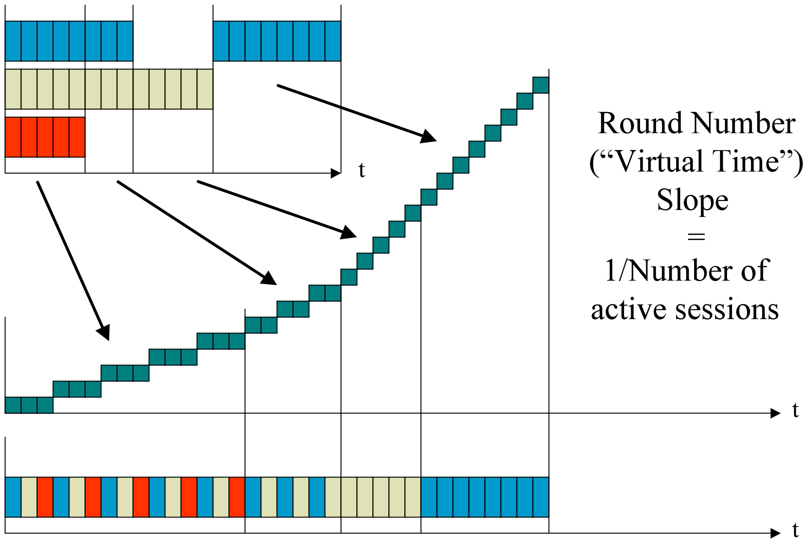
Bit Round Robin Scheduling



Bit Round Robin Scheduling

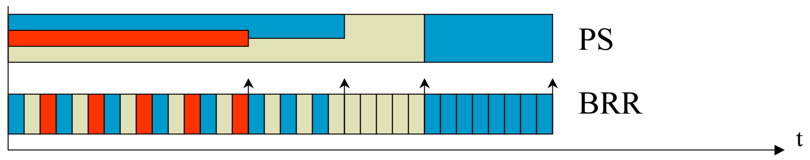


Bit Round Robin Scheduling



Generalized Processor Sharing

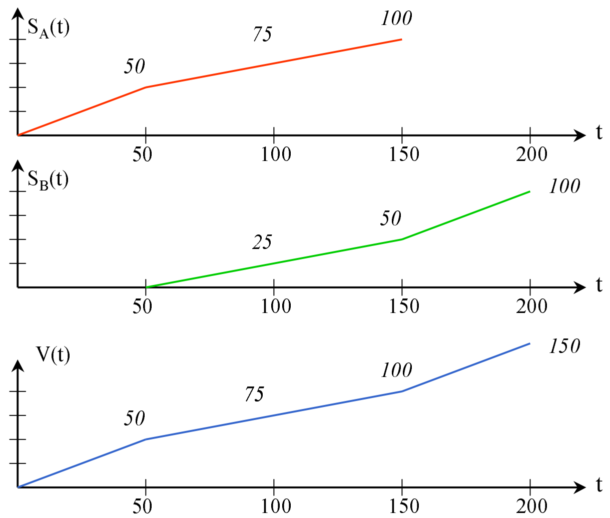
- Can add *weights* to each flow:
  - A flow with weight "2" gets to send two bits per round.
- Processor Sharing - send a small fraction of a bit each round
  - Like having multiple flows *sharing* the output line at the same time.
  - No flow has to wait for "its turn".
  - Is perfectly fair at all times.



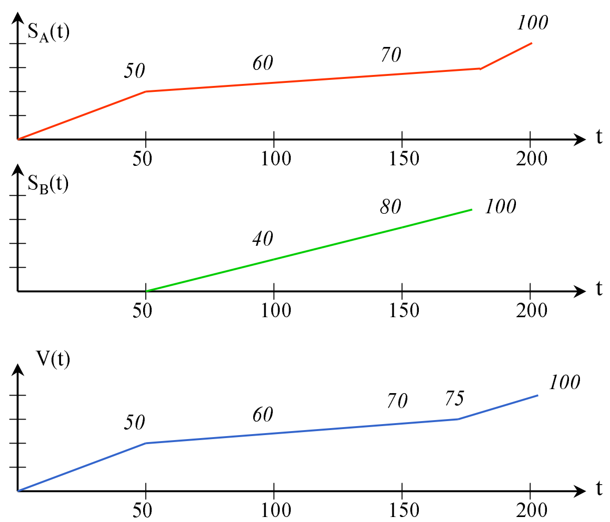
PS - Another Example

- Packet A arrives at time 0
- Packet B arrives at time 50
- Both packets are 100 bits long.
- Output line rate is 1 b/s.

PS - Another Example



GPS - A Weighted Example



$W_A = 1$

$F_A = V(0) + L / W_A$
$= 0 + 100 / 1$
$= 100$

$W_B = 4$

$F_B = V(50) + L / W_B$
$= 50 + 100 / 4$
$= 75$

Fair Queuing

- Cannot implement GPS
  - No way to serve more than one flow at a time.
  - Would rather not break packets to small pieces (overhead).
  - No real discipline can be as fair as GPS: when flow A is served we are unfair to all other flows.
- Can try to emulate GPS
  - Bounded approximation error.
  - Keep implementation cost low.
- "Weighted Fair Queuing", *S. Keshav 1989*.
  (Packet GPS in Parekh terminology)

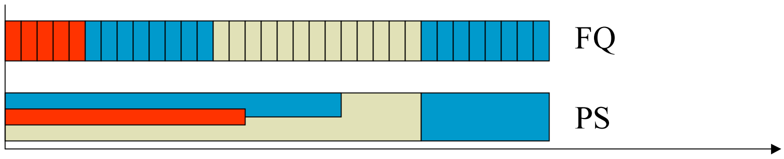
FQ - Basic issues

- Basic Idea
  - Serve packets in order of their *finish time* had we been doing GPS.
  - Cannot always do this unless we know the future.
- Implementation
  - Key idea: Round number (Virtual time) is an increasing function of time.
  - Can calculate finish number of packet when it arrives.
  - Send packets in order of their finish numbers.
  - Need to keep track of current round number V(*t*).

FQ - Example

- Note that FQ (unlike packet by packet round robin) handles variable packet sizes.

Packets are sent in order of departure in the corresponding PS system



FQ - Implementation

- When a packet of flow *f* arrives calculate its finish round number:
  - If flow *f* is active, $F(f, k) = F(f, k\text{-}1) + L(k)$
  - Otherwise, $F(f, k) = V(t) + L(k)$
- Need to keep track of virtual time $V(t)$
  - Slope of $1/N(t)*r$
  - Slope may change whenever a packet arrives (in GPS) or departs.
- Must keep track of active sessions in the simulated GPS systems
  - $Next(t) = t + [\ F_{min} - V(t)\ ]N(t)*r$
  - Computationally expensive.

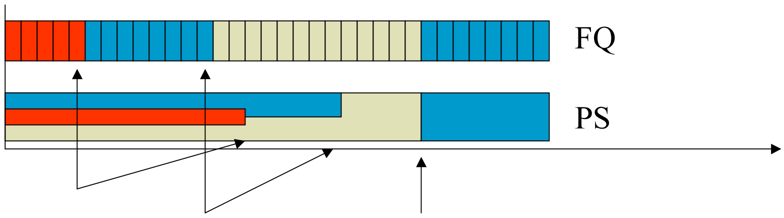
FQ - Implementation

- When the line becomes idle, transmit the packet with the *smallest* finish number.
  - V(t) is an increasing function of time.
  - If $F_A > F_B$, GPS will finish sending packet A *after* packet B.

WFQ and GPS

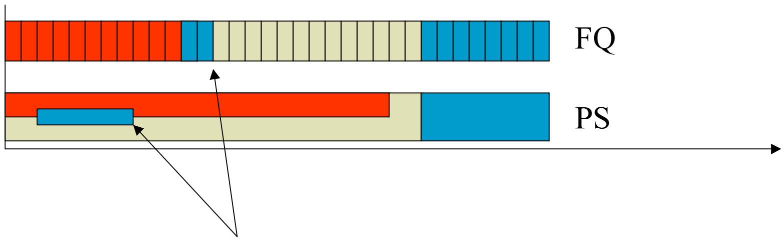
- GPS is not ahead of WFQ by more than one packet (in terms of transmitted bits up to time *t*).
- Packets in WFQ are not delayed more than one packet relative to GPS:
  - $S_{GPS}(f, t) - S_{WFQ}(f, t) \leq L_{max}$
  - $D_{WFQ}(f, k) - D_{GPS}(f, k) \leq L_{max} / r$

Delay Bound



When WFQ and GPS finish packets at the same order,
WFQ will never lag behind GPS.
(And may sometimes be ahead of GPS).

Delay Bound



When a packet arrives too late, it may have to wait for the current packet to finish transmission, hence L_{max}/r delay.

- Error term does *not* accumulate over time.
- May accumulate over multiple network hops.

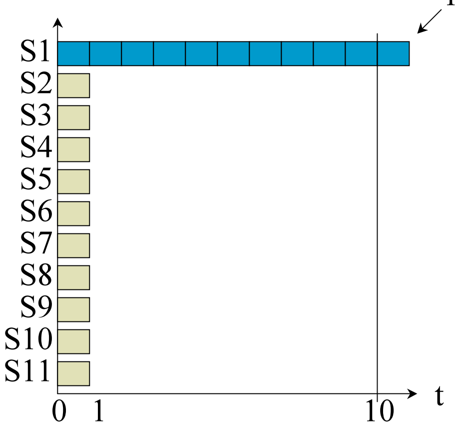
Worst Case WFQ (WF^2Q)

- Contrary to popular belief WFQ does *not* approximate GPS to within a difference of one packet.
  - No lower bound:
    $S_{WFQ}(f, k) - S_{GPS}(f, k) \leq L_{max}$
  - In fact WFQ might be well *ahead* of GPS!

- Less delay, but..
- More *jitter*,
- Difficult to estimate available bandwidth.
  - Necessary for best-effort traffic.

W^2FQ - Example

11th packet

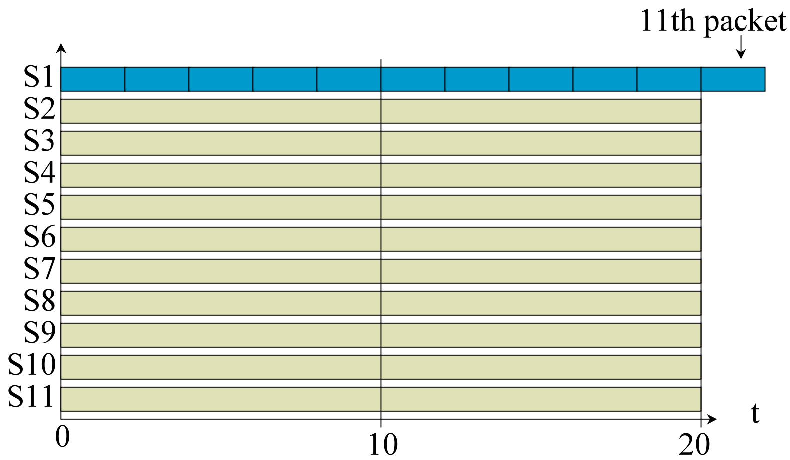


R1=0.5

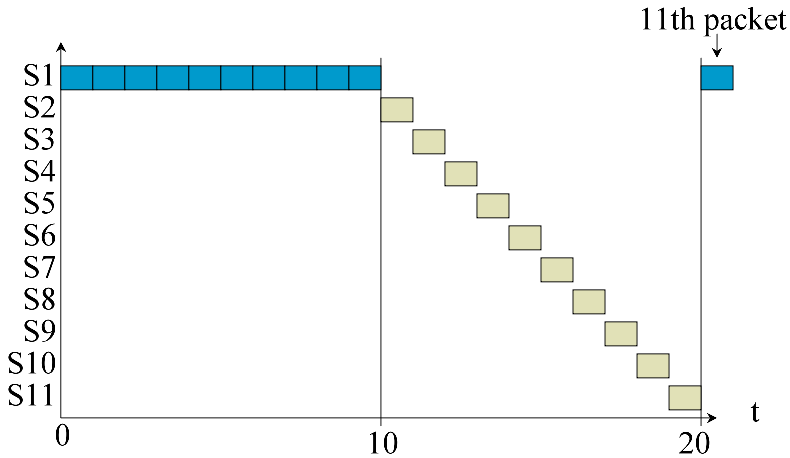
R2=R3=…=R11=0.05

Packet lengths: 1 sec

GPS Service Order



WFQ Service Order



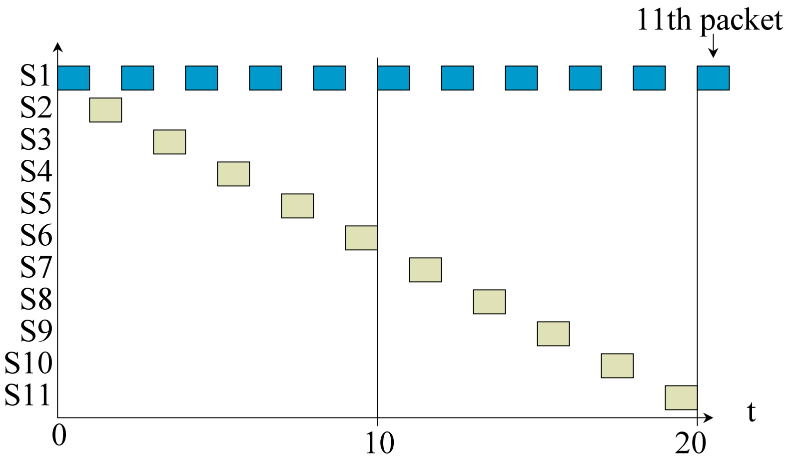
WF²Q Basics

- In WFQ the scheduler selects next packet with minimal finish number, *among all available packets*.
- To minimize difference between packet system and fluid-GPS, scheduler should consider *only packets that have started* in the emulated GPS system.

$$S_{i,GPS}(0,\tau) - S_{i,WF^2Q}(0,\tau) \le L_{\max}$$

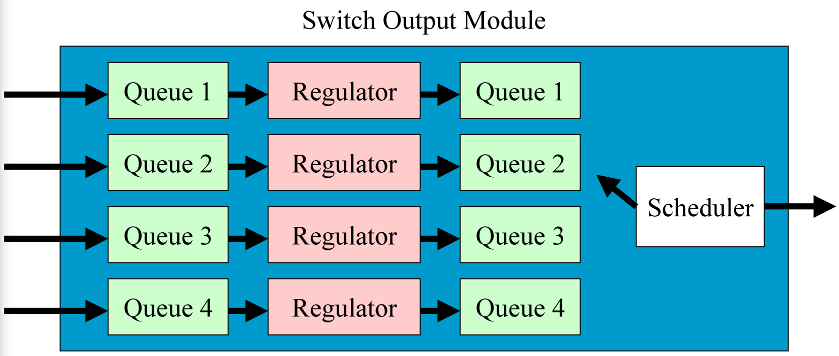
$$S_{i,WF^2Q}(0,\tau) - S_{i,GPS}(0,\tau) \le \left(1 - \frac{g_i}{r}\right) L_{i,\max}$$

W²FQ Service Order



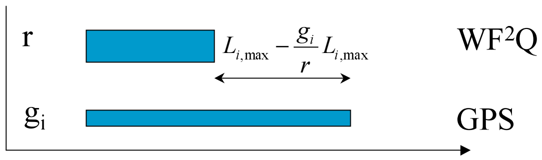
Implementation: Rate Controlled Scheduling

- Regulator holds packets until they are *eligible* for transmission.
- Scheduler decides which eligible packet should be transmitted next.



WF^{2}Q Properties

- Discipline is *work-conserving:*
  - Rate controller + GPS scheduler is the same as GPS alone.
  - Replace GPS with WFQ (both work conserving) to get WF^{2}Q.
- Lower bound
  - WF^{2}Q starts sending packet no earlier than GPS.
  - GPS may take more time to transmit the packet (depending on allocated and available bandwidth).



What next?

- WFQ (or WF^2Q) approximate GPS service to within a constant error term.
- GPS is much simpler to analyze (fluid model).
- Strategy
  - Determine performance bounds for networks of GPS schedulers.
  - Use our previously derived bounds to bound the performance of a corresponding WFQ network.

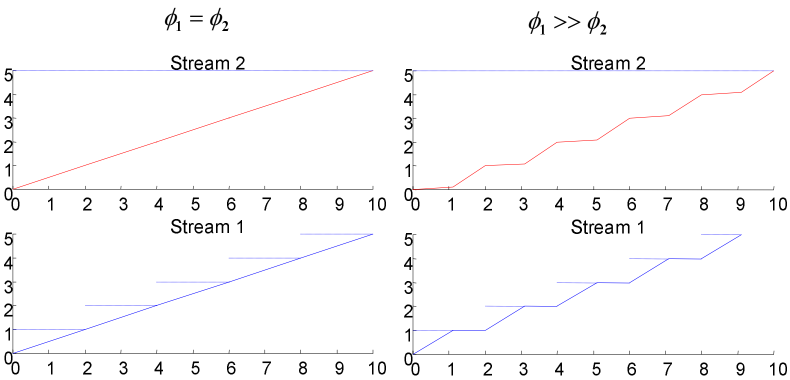
GPS Performance Analysis

Analyzing Networks of GPS schedulers.

GPS - Mathematical Model

- Generalized Processor Sharing is defined by $\frac{S_i(\tau,t)}{S_j(\tau,t)} \geq \frac{\phi_i}{\phi_j}$
  - Assuming session *i* is backlogged.
- Session 'i' is guaranteed a rate of $g_i \geq \frac{\phi_i}{\sum_j \phi_j} r$
- Protected from other sessions.
- Can reduce delays for a session by increasing its weight ϕ.
  - Corresponding increase in delay for other sessions.
  - Less impact on other sessions when the better treated session is *steady*.

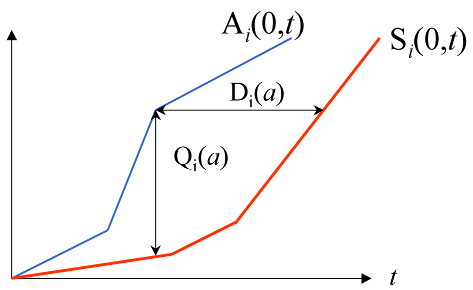
Weights Allocation



- Session 1 is a steady source (delay sensitive real time traffic).
- Session 2 is a bursty source (best effort traffic).

Guaranteed Performance

- Delay depends on arrival functions of *all* sessions.



- May also help bound the maximum queue length (I.e, size of buffers needed by the switch for every session).

Networks of GPS Servers

- Assume session '*i*' is leaky bucket constrained:
  - $A_i \sim (\sigma_i, \rho_i)$
  - Other sessions are not constrained.
  - Network may not even be stable.

- Assume g_i is the minimum guaranteed bandwidth allocated to session '*i*' along its path.
  - Session is *locally* stable: $g_i \geq \rho_i$

- Session '*i*' can be guaranteed maximum delay: $D_i \leq \frac{Q_i^{max}}{g_i} \leq \frac{\sigma_i}{g_i}$

WFQ End-to-End Delay

- When the servers are WFQ:

$$D_i^* \le \frac{\sigma_i}{g_i} + K\frac{L_{\max}}{g_i} + K\frac{L_{\max}}{r}$$

- First error term due to store & forward delay
  - Packets are processed by the scheduler when their last bit arrives.
- WFQ error term
  - Negligible when line rate is high.

A Simple Example

- Given:
  - A constrained connection A ~ (16 KByte, 150 Kbps)
  - 10 network hops (rate 45 Mbps)
  - Packet size are up to 8 KB
  - Total propagation delay: 30 ms.
- What is the required *guaranteed bandwidth* to get an end-to-end delay not more than 100 ms?
- Solution: 12.87 Mbps!
  - S&F delay contributes 46 ms to delay!
  - All other terms: 24 ms

Leaky Bucket Constrained Sources

- To reduce delays for a given session, we must increase its weight.
  - Increases guaranteed bandwidth for this session.
  - Reduces network capacity.
  - May overload the network turning it unstable.
- When is it possible to reduce delay while still maintaining stability?

Consistent Relative Session Treatment (CRST)

- Assume all sources are leaky bucket constrained $A_i \sim (\sigma_i, \rho_i)$
- **Definition:**
  - Session *j* is said to *impede* session *i* at node *m* if:

$$\frac{\phi_j^m}{\rho_j} > \frac{\phi_i^m}{\rho_i}$$

- **Definition:** Consistent Relative Session Treatment (CRST)
  - A GPS weight assignment for which:
  - If session *i* impedes session *j* at node *m*, then session *i* impedes session *j* at any other node where they contend for bandwidth.

Stability Condition

- **Theorem:**
  - A CRST GPS network is stable if the utilization at every node is less than one.

$$\sum_{i \in \{\text{sessions in node } m\}} \rho_i < r^m$$

- Finding D* and Q* for a general GPS network is a complex optimization problem.
- It turns out that it is easily bounded for CRST networks.
- In the following slides we will sketch an outline of the proof.

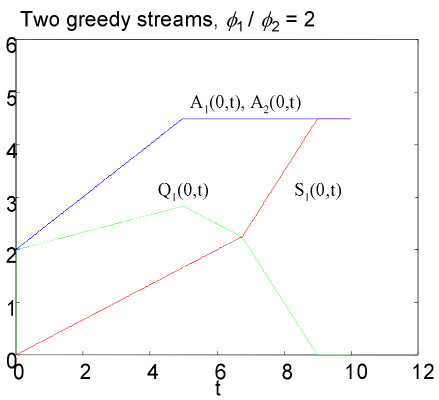
The Single Node Case

- Assume all sources are leaky bucket constrained $A_i \sim (\sigma_i, \rho_i)$
- **Definition**: Greedy Source
  - A leaky bucket constrained source is greedy starting at time t if its arrival function is maximal for all time $\tau > t$.
  - Uses all available tokens and maximum rate ρ_i
- **Theorem:**
  - For every session D^* and Q^* (worst delay and backlog) are achieved when *all* sessions are greedy starting at time 0, the beginning of a system busy period.

Greed...

- Intuitively simple:
  - When all sessions are greedy starting at time 0, *maximum* traffic is entering the network in (0,*t*).
  - All sessions are active, therefore each one is allocated *minimal* guaranteed bandwidth.
  - Result: maximum backlog and delay.
- No need to solve a difficult optimization problem to determine Q* and D*.

Single Node - Delay and Backlog



- Note that D* and Q* may not be achieved at the same time!
  - D depends on backlog size and clearing *rate*.

Feasible Ordering

- Assume all sessions are greedy starting at time 0.
- Server will finish serving all backlog at time:

$$t_B = \frac{\sum_{i=1}^{N} \sigma_i}{r - \sum_{i=1}^{N} \rho_i}$$

- Number the sessions according to the order in which their backlog is cleared.
- For the first session: $\rho_1 \le \frac{\phi_1}{\sum_{j=1}^{N} \phi_j} r$
- The second: $\rho_2 \le \frac{\phi_2}{\sum_{j=2}^{N} \phi_j} (r - \rho_1)$

- There may be more than one possible feasible ordering.
- The one that comes into play at time 0 depends on the σ_i's.

An Important Inequality

- **Theorem:**
  - Let 1,..,N be a feasible ordering, then for any time *t* and session *p*:

$$\sum_{k=1}^{p} \sigma_k^t \le \sum_{k=1}^{p} \sigma_k$$

  - I.e, burstiness is not increased by the GPS server.

Output Burstiness

- Maximum output burst (for session *i*) results when:
  - Backlog is maximal (Q^*)
  - Session *i* is the only active session (highest output rate).

$$\sigma_i^{out} \geq Q_i^*$$

- Other direction:

$$S_i(\tau,t) \leq Q_i(\tau,t) + \rho_i(t-\tau) \leq Q_i^* + \rho_i(t-\tau)$$

$$\sigma_i^{out} \leq Q_i^*$$

- Calculate Q* and determine output burstiness.

Network Internal Burstiness

- Following Cruz, we compute the burstiness at the output of every node on the session's path.
- If session *i* does not impede session *j* at node *m* then σ_j^{out} is independent of σ_i^{out}
- Iterative procedure computes σ^{out} for all sessions at all nodes
  - Characterizes traffic entering every network node.

Computing Bounds for Networks with Known Internal Burstiness

- Maximum delay is achieved when all sessions entering a node are greedy at the same time, *but*…
- Network topology may preclude certain arrival functions.
- Should compute maximum delay over all *possible* arrival functions of all sessions at all nodes.
  - Difficult optimization problem.

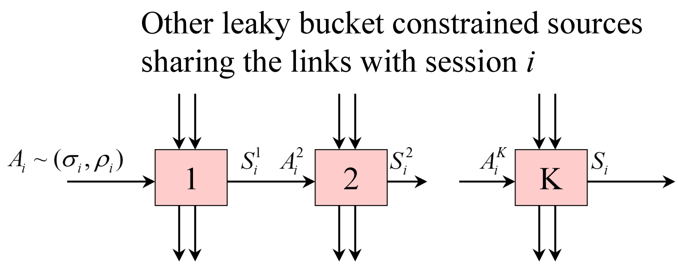
Independent Sessions Relaxation

- When computing session *i*'s delay and backlog assume all *others* sessions (*independent* sessions) can send traffic at (σ_j^m, ρ_j).
- Session *i* traffic is constrained to flow along its route so that:

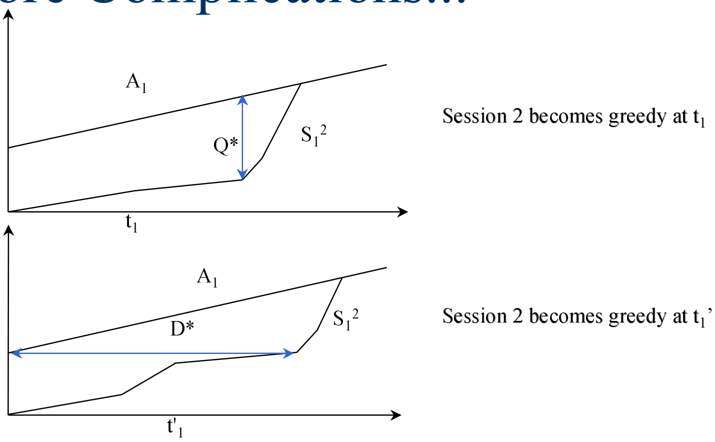
$$A_i^m = S_i^{m-1}$$

- Upper bounds the delay and backlog of session *i*.
  - Every arrival function allowable in the network is allowed under the relaxation too.

Independent Sessions



More Complications...

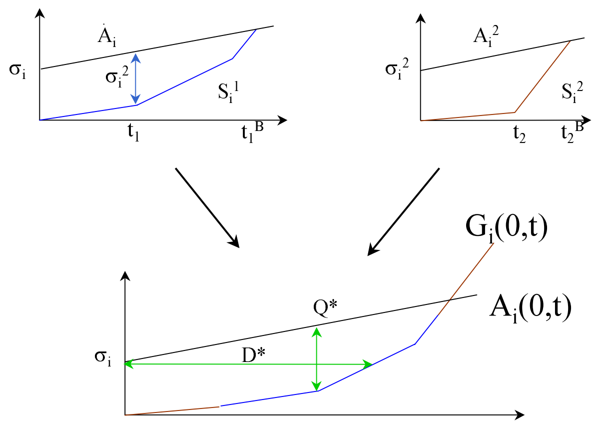


- Q* and D* are not necessarily achieved with the same arrival functions.

The Universal Service Curve

- We are looking for:
  - *End-to-end* delay (D*)
  - $Q^* = Q^1 + Q^2 + \ldots$ (total session *i* traffic in the network)
- Crucial insight:
  - Session *i* will always be limited by the *slowest* link on its path.
  - The set of all feasible rates for session *i* on link *m* is given by the slopes of $S_i^m(0,t)$ (assuming all sessions are greedy).
- Solution:
  - Construct the universal curve $G_i(0,t)$ by concatenating the segments of all $S_i^m(0,t)$ in increasing slope order.

Universal Curve - An Example



- The Universal Curve makes calculating D* and Q* easy!

Summary - GPS Networks

- If all sources are known to be leaky bucket constrained,
  - And we limit ourselves to CRST networks…
  - We can control each session's delay by adjusting the weights at each GPS node.
  - We can easily bound the delay and backlog.
- If only some sources are leaky bucket constrained,
  - Bandwidth and delay are coupled.
  - Can reduce delay for a session only by increasing its allocated bandwidth, thereby wasting network resources!
  - GPS protects sessions from bandwidth hogs.

Integrated Services in the Internet

Resource Reservation Protocol (RSVP)

Signaling

- Users need to communicate their bandwidth / delay requirements to the network.
  - Call setup.
  - Connection teardown.
  - Renegotiation.

- Switches communicate to select a path with enough resources.

Real World Use - RSVP

- Each router implements WFQ or other per-flow scheduling discipline.
- Resource Reservation Protocol used to request bandwidth allocation.

- Source specifies *Flowspec* (σ_i, ρ_i).
- Destination requests bandwidth (affects delay).

- Reservation request propagates from destination to source allocating bandwidth in each link (assigning *weights*).
- If bandwidth is not available, request is denied (*admission*).

Scaling

- Needs a QoS-capable routing protocol to select paths with available bandwidth.
  - More information to distribute through the network.
  - More changes as flows are started and destroyed.
- All routers must store per-flow information.
  - Less robust.
  - Difficult to implement at high speed (Internet backbones).

Aggregation

- More aggregation:
  - Less sessions to consider in backbone switches.
  - Less signaling.
  - BUT: less isolation!
- Solution:
  - Aggregate to a *class*.
  - Members of the same class have similar performance requirements.
  - Police traffic at the edge of the network (no protection inside network between sessions of the same class).

The End

Work Conserving vs. Non Work Conserving Service Disciplines

- Work conserving scheduler is *never* idle when there are packets in the queue.
- GPS, WFQ, WF^2Q are all work conserving disciplines.
- Non work conserving schedulers hold packets in queue until their eligibility time is reached (e.g., TDM).

Non Work Conserving Disciplines

- Reduce network internal burstiness.
  - Smaller buffers needed at switches.
- Increases mean delay.
  - Not a problem for *playback* applications.
  - Does not hurt worst-case performance.
- Wastes bandwidth.
  - Can serve best effort traffic when idle.
- High implementation costs at every switch (regulators).