Design Issue for Integrated Services Networks

Design Issue for Integrated Services Networks

- Difference between traditional best-effort data applications and real-time audio and video services:
  - Best-effort applications are elastic; real-time applications are not
  - Best-effort applications are congestion-avoidant; real-time applications are congestion-ignorant
- Existing internet supports only best-effort services
- Two possible approaches to designing integrated services networks:
  - Change the network and application implementation, but not the architecture
  - Change the architecture

Change the Implementation Approach

- Use packet scheduling algorithms in routers to eliminate the interference of real-time applications with data applications
- Address lack of elasticity at the application level
- Pros:
  - No changes required at the network interface $\Rightarrow$ incremental upgrade
  - Relatively easy to deploy
- Cons:
  - Network delivers the same class of service to all applications $\Rightarrow$ no assurance of QoS
  - Application development is tricky

Change the Architecture Approach

- Extend the service model to include a wide range of services
- Pros:
  - Easier application development, appropriate service abstraction
- Cons:
  - Requires fundamental changes in existing networks $\Rightarrow$ difficult to deploy
- Fundamental questions:
  - Why should we extend the service model?
  - Who selects the service for a flow?
  - Do we need admission control?

Why Extend the Service Model

- Goal of network design: maximize the performance of resident applications
- Need for extending the service model:
  - Example: Exponential server with rate $\mu = 1$; two flows with Poisson arrivals with rates $r = 0.25$; $U_1 = 4 - 2d_1$ and $U_2 = 4 - d_2$
  - System can be modeled as an M/M/1 queue
  - FIFO service: $d = \frac{1}{\mu - r} \Rightarrow V = 2$
  - Priority service: $d_1 = \frac{1}{\mu - r_1}$; $d_2 = \frac{\mu}{(\mu - r_1)(\mu - r_1 - r_2)} \Rightarrow V = 8/3$
- Statistical sharing of resource yields increased efficiency $\Rightarrow$ applications should be mixed on a network

Who Selects the Service for a Flow

- Implicitly supplied
  - Pros: no changes to the service interface; service does not need to be uniform across routers
  - Cons: fixed set of application classes; need for network to understand application requirements (e.g., conferencing vs. lecture)
- Explicitly requested
  - Network supports a set of service classes and the applications select the ones that they want
  - Pros: Clean separation between network and applications
  - Cons: Incentive/pricing issues need to be addressed
  - Need for stable, but extensible, service model

Need for Admission Control

- Overloading: state in which the overall efficacy of the system increases with decrease in the number of flows
- Preventing overloads:
  - Overprovisioning
    * If cost-effective, then this is simpler to do
    * Depends on the cost of overprovisioning and application requirements
  - Admission control: avoid overloading by restricting admission
    * New paradigm

Integrated Services Networks - Summary

- Support applications with diverse **Quality of Service** (QoS) requirements
- Provide QoS guarantees
  - Bandwidth
  - Packet delay and loss
- Provide best-effort service
- Most significant changes required at the network layer
- Fundamental problem:
  - **Control congestion**: meet QoS requirement

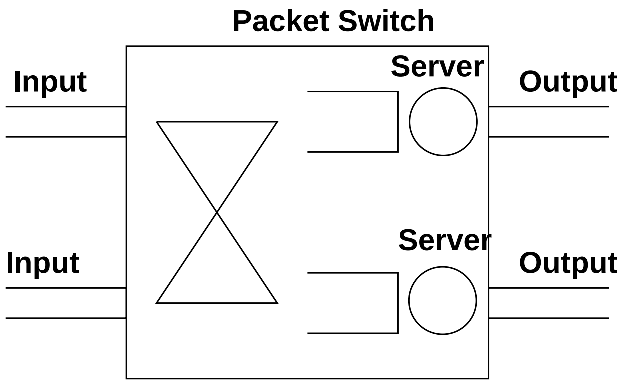
Congestion Control

- **Reactive**:
  - End-to-end feedback-based control
  - Simple network layer, but does not provide any performance guarantees
- **Preventive**:
  - Source specifies its traffic characteristics and QoS requirements (in terms of delay, jitter, bandwidth, and packet loss)
  - Network employs a *packet scheduling* algorithm - which determines order of packet transmission
  - Contract: network guarantees to meet the performance requirements as long as the source conforms to the specifications

Packet Scheduling

Network Model

- Connection oriented network layer
  - Packets traverse same path
  - Flow $\equiv$ connection
- Output buffered packet switch



Packet Scheduling Algorithm

- Fundamental mechanism to provide different QoS to different flows
- Packet scheduler:
  - Determines order of packet transmission
  - Controls packet delay and loss for a flow
- Examples: Virtual Clock, WFQ, Delay EDD, etc.

Review: Real-time Scheduling

- Process Model (c_i, p_i, d_i)
- Assume $p_i = d_i$
- Earliest Deadline First Scheduling
- Deadline $= AT_i(k) + d_i$
- Schedulability Test:

$$\sum_{j=1}^{j=N} \frac{c_j}{p_j} \leq 1$$

Limitations and Modifications

- Problem:
  - EDF scheduling does not protect flows from each other when sources do not conform to the specification (e.g., a source sending packets at a higher rate than negotiated)
- Solution:
  - Assign a rate to each flow and compute deadline based on expected arrival time

$$EAT_i(k) = EAT_i(k-1) + \frac{l_i}{r_i}$$

$$d_i = p_i = \frac{l_i}{r_i} \quad c_i = \frac{l_i}{C}$$

  - Schedulability test:

$$\sum_{j=1}^{j=N} \frac{c_j}{p_j} \leq 1 \Rightarrow \sum_{j=1}^{j=N} \frac{r_i}{C} \leq 1$$

- Problem: Source may send too slow and then very fast, leading to violations of QoS for other flows

Delay and Jitter EDD Algorithms

- Approach: Reset expected arrival time with packet arrivals

$$EAT_i(k) = \max\{EAT_i(k-1) + \frac{l_i}{r_i}, AT_i(k)\}$$

- Delay EDD:
  - Schedule packets in the order of $EAT_i(k) + d_i$
  - Separates rate and delay allocations
- Jitter EDD:
  - Hold packet until $EAT_i(k) = \max\{EAT_i(k-1) + \frac{l_i}{r_i}, AT_i(k)\}$.
- Comparison of Delay-EDD and Jitter-EDD
  - If K network switches the maximum end-to-end delay is Kd in both the algorithms
  - Minimum end-to-end delay is 0 for Delay-EDD and $(K-1)d$ for Jitter-EDD
  - Buffer requirement for Delay-EDD is higher than for Jitter-EDD.

Virtual Clock Scheduling Algorithm

- Virtual Clock scheduling algorithm:
  - Assign a rate r_i to each flow
  - Assign a Virtual Clock (VC) value to each packet as follows:

$$EAT_i(k) = \max\{EAT_i(k-1) + \frac{l_i^{k-1}}{r_i}, AT_i(k)\}$$

$$VC_i(k) = EAT_i(k) + \frac{l_i^k}{r_i}$$

  - Schedule packets in the order of their VC values
- Delay guarantee: If $\sum_{j=1}^{j=N} r_j \leq C$, then a packet is transmitted by a VC scheduler by $VC_i(k) + \frac{l^{max}}{C}$

Discussion

- Jitter EDD is non-work conserving
- Delay EDD and Virtual Clock scheduling algorithms:
  - Work conserving
  - Allow sources to utilize idle bandwidth; but later penalize them (i.e., they are *unfair*)
- Need fair scheduling algorithms that are work conserving

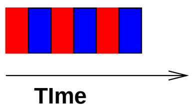
Fair Algorithms

- Flows assigned weights
- Fairness: flows receive bandwidth in proportion to weights

$$\frac{W_f(\Delta)}{r_f} - \frac{W_m(\Delta)}{r_m} = 0$$

$W_f(\Delta)$	:	service received by flow f in Δ
r_f	:	weight of flow f

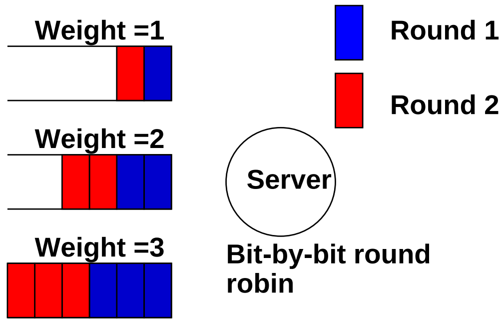
- Packet-by-packet service: $\left| \frac{W_f(\Delta)}{r_f} - \frac{W_m(\Delta)}{r_m} \right| \leq H(f,m)$



- $H(f,m)$: unfairness measure
- Objective: achieve small unfairness measure

Weighted Fair Queuing (WFQ)

- Emulates bit-by-bit weighted round-robin server



- Schedule packets in the finish order in the round-robin server

WFQ Implementation

- Start tag $S(p^j)$: round in which packet begins service

$$S(p^j) = \max\{v(A(p^j)), F(p^{j-1})\}; j \geq 1$$

$$\frac{dv(t)}{dt} = \frac{C}{\sum_{i \in B(t)} r_i}$$

- Finish tag $F(p^j)$: round in which packet finishes service

$$F(p^j) = S(p^j) + \frac{l^j}{r}$$

p^j	:	j^{th} packet of flow
l^j	:	length of p^j
r	:	weight of flow

- Packets scheduled in increasing order of finish tag
- Delay property: If $\frac{\phi_i}{\sum_{j \in A} \phi_j} C \geq r_i$, then a packet is transmitted by $EAT_i(k) + \frac{l^{max}}{C}$
- Problem: implementation is computationally expensive

Self Clocked Fair Queuing (SCFQ)

- Assigns two tags - start tag S and finish tag F - to each packet

$$S_i(j) = \max\{F_i(j-1), v(AT_i(j))\}$$

$$F_i(j) = S_i(j) + \frac{l_i^j}{\phi_i} \quad j \geq 1$$
$$F_i(0) = 0$$

- The virtual time at time t, $v(t)$, is defined to be equal to the start tag of the packet in service at time t. $v(t) = t$ when the server is idle.

- Schedule packets based on their finish tags

Properties of SCFQ

- SCFQ has Bounded Unfairness: A channel does not receive $2l^{max}$ more than its fair share.

- Delay guarantee: If $\phi_j = r_j$ and $\sum r_j \leq C$, then a packet is transmitted by

$$EAT_i(k) + \sum_{j \in A \wedge j \neq i} \frac{l_j^{max}}{C}$$

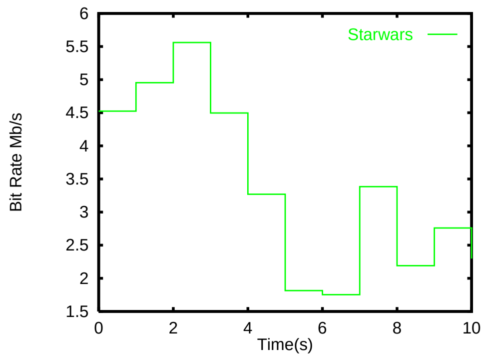
Classification of Packet Scheduling Algorithms

	Rate Allocation	Delay Allocation
Work Conserving	Virtual Clock Packet-by-packet GPS (PGPS) Self Clocked Fair Queuing (SCFQ)	Delay EDD
Non-work Conserving	Hierarchical Round Robin Stop-and-Go Queuing	Jitter EDD Rate Controlled Static Priority Queuing

Selecting a Packet Scheduling Algorithm

Application Requirements

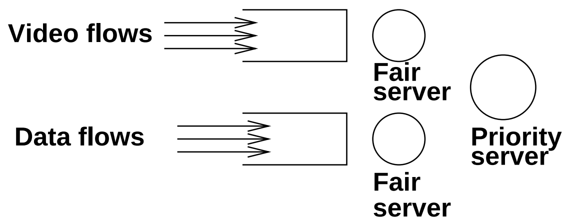
- Video:
  - Unpredictable and highly variable bit rate requirement



  - Unpredictability: idle resource usage must not be penalized
  - Variability: QoS guarantees in overload
  - Fair scheduler desirable

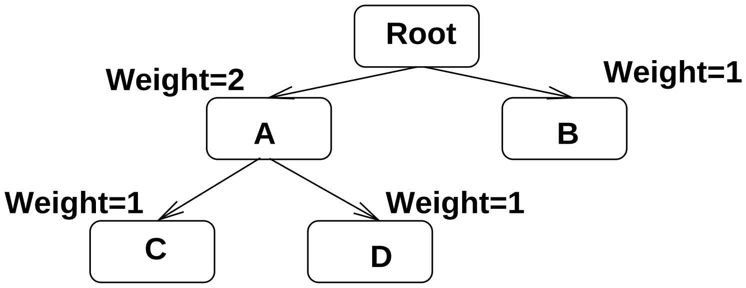
Application Requirements (Cont'd)

- Audio: low worst-case packet delay
- Data
  - Interactive: low average packet delay
  - Throughput intensive:
    * Decentralized, heterogeneous environments: fairness
    * Coexistence with VBR video: fairness over variable rate servers



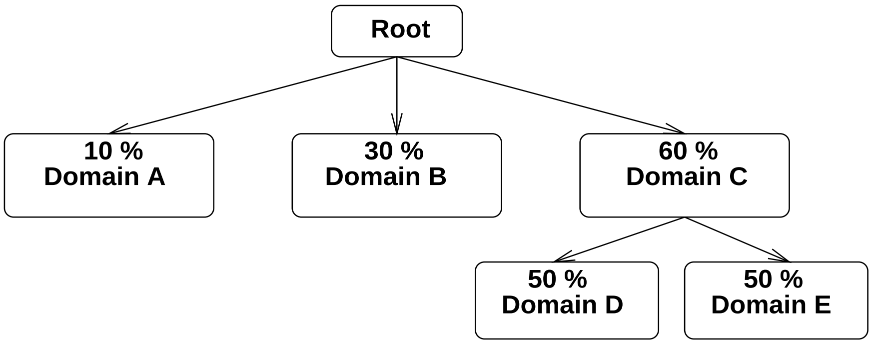
Network Requirements

- Heterogeneity in services and protocols: hierarchical link bandwidth allocation
  - Isolation between different service classes
    * Different services can be designed independently
    * Different scheduling algorithms for different service classes



Network Requirements (Cont'd)

- Multiple administrative domains: hierarchical link bandwidth allocation



- High speed networks: efficient

Requirements Summary

- Fairness for VBR video
- Fairness over variable rate servers for throughput intensive flows
- Low average and worst-case packet delay for low throughput flows
- Hierarchical link bandwidth allocation
- Computationally efficient
- **Objective**: scheduling algorithms that meet these requirements

Weighted Fair Queuing (WFQ)

- Start tag $S(p^j)$: round in which packet begins service

$$S(p^j) = \max\{v(A(p^j)), F(p^{j-1})\}; j \geq 1$$

$$\frac{dv(t)}{dt} = \frac{C}{\sum_{i \in B(t)} r_i}$$

- Finish tag $F(p^j)$: round in which packet finishes service

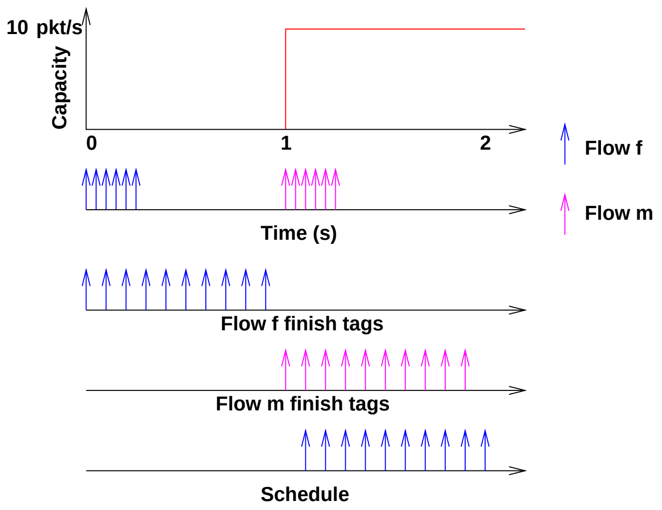
$$F(p^j) = S(p^j) + \frac{l^j}{r}$$

p^j	:	j^{th} packet of flow
l^j	:	length of p^j
r	:	weight of flow

- Packets scheduled in increasing order of finish tag

Limitations of WFQ

- Emulation computationally expensive
- WFQ unfair over variable rate servers



Limitations of Existing Algorithms

- Fairness over variable rate servers required for hierarchical link allocation
- Conclusion: WFQ unsuitable for integrated services networks
- Other algorithms: either are unfair over variable rate servers or have unsuitable delay properties

Key Observations

- Algorithm should be defined independent of server capacity to achieve fairness over variable rate servers
- Schedule packets in start tag order to provide low delay to low throughput flows
  - Finish tag inversely proportional to weight

$$F(p^j) = S(p^j) + \frac{l^j}{r}$$

  - Scheduling in finish tag order: low delay to high throughput flows

Department of Computer Science, UMass Amherst

Start-time Fair Queuing (SFQ)

- Each packet stamped with start tag $S(p^j)$ and finish tag $F(p^j)$

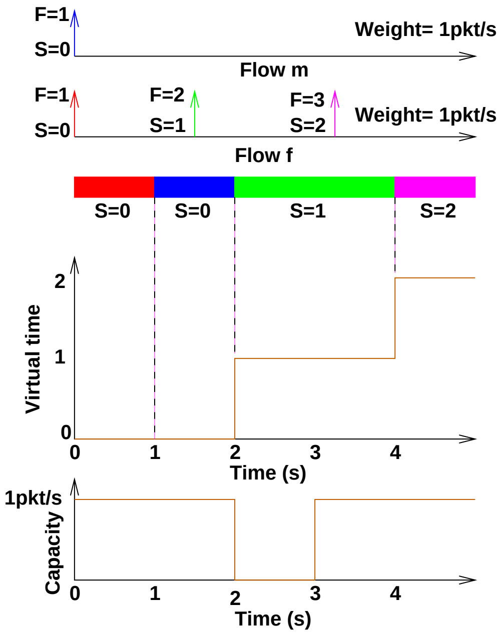
$$S(p^j) = \max\{v(A(p^j)), F(p^{j-1})\}$$

$$F(p^j) = S(p^j) + \frac{l^j}{r}$$

$A(p^j)$: arrival time of packet
$v(t)$: virtual time

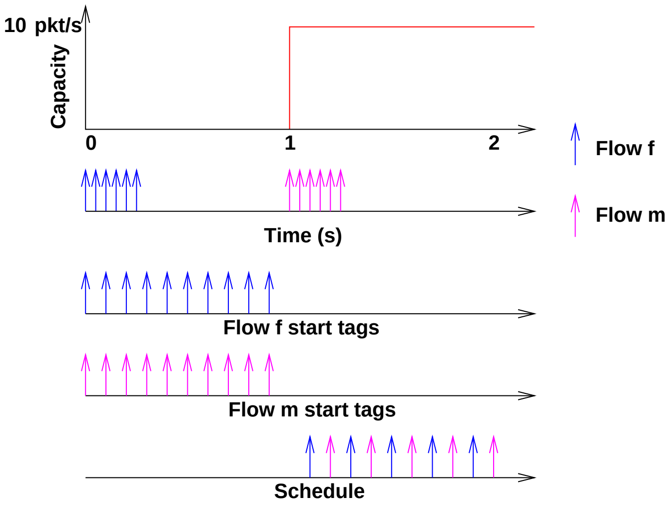
- Virtual time $v(t)$: start tag of the packet in service at time t
- Packets scheduled in increasing order of *start tag*

SFQ: An Example

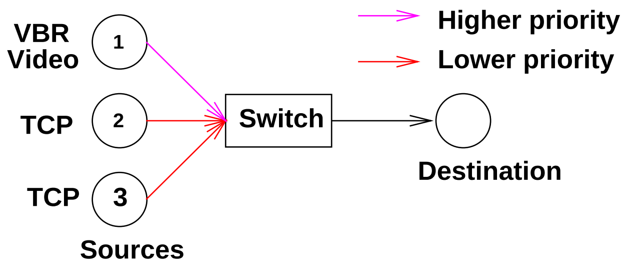


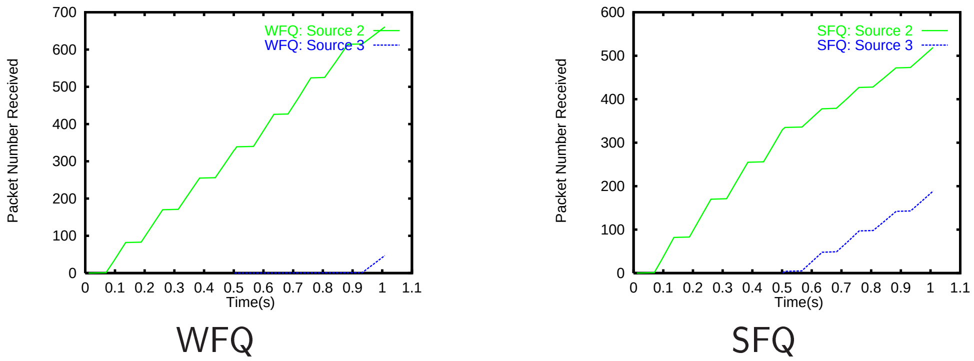
Fairness of SFQ

- SFQ fair over variable rate servers



Fairness: Experimental Evaluation



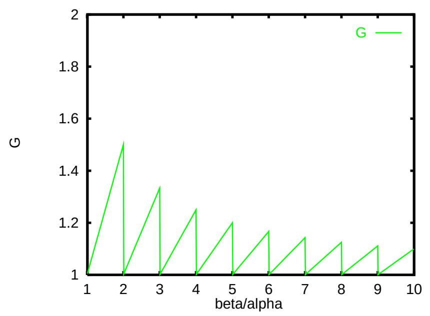


Fairness Guarantee

- **Theorem**: If flow f and m backlogged in Δ:

$$\left|\frac{W_f(\Delta)}{r_f} - \frac{W_m(\Delta)}{r_m}\right| \leq \left(\frac{l_f^{max}}{r_f} + \frac{l_m^{max}}{r_m}\right)$$

- Goodness metric: $G = \left(\frac{l_f^{max}}{r_f} + \frac{l_m^{max}}{r_m}\right) / L(f, m)$
  - On average G is within 11% of the lower bound



Server Model

- Fluctuation in server rate:
  - Broadcast and wireless links
  - Variation in processing bandwidth
- Fluctuation Constrained (FC) server:

$$T(w) \leq \frac{w}{C} + \delta(C)$$

C	:	server capacity
$\delta(C)$	:	burstiness
$T(w)$	:	time to serve packets of aggregate length w

- Exponentially Bounded Fluctuation (EBF) server: probabilistic relaxation of FC server

Department of Computer Science, UMass Amherst

Deadline Guarantee

- Objective: determine deadline guarantee for sources with different characteristics
- Deadline guarantee based on expected arrival time of packet
  - This guarantee is called *delay guarantee*
- Expected arrival time of a packet: the time at which the packet begins service in a server of rate r

$$EAT(p^j) = \max \left\{ A(p^j), EAT(p^{j-1}) + \frac{l^{j-1}}{r} \right\}$$

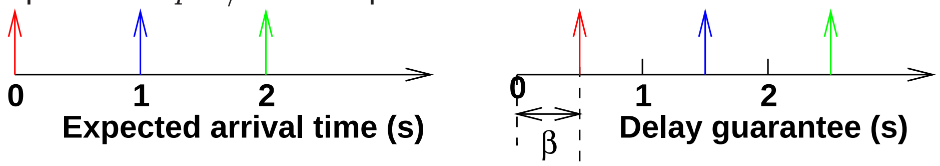
Delay Guarantee

- **Theorem**: In a FC SFQ server

$$L_{SFQ}(p^j) \leq EAT(p^j) + \beta^j$$

$$\begin{array}{lcl} L_{SFQ}(p^j) & : & \text{departure time of } p^j \\ \beta^j & : & \sum_{n \in N \wedge n \neq f} \frac{l_n^{max}}{C} + \frac{l^j}{C} + \delta(C) \end{array}$$

- Example: $r = 1pkt/s$ and 3 packet sent at $t = 0$

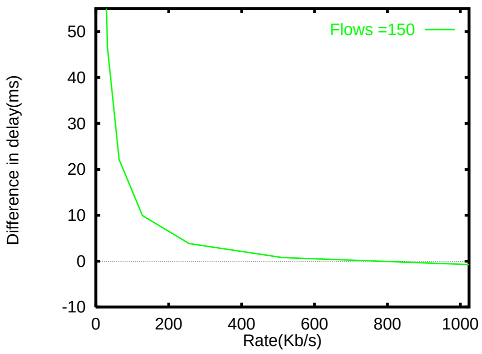


- Delay guarantee governed only by flow's behavior
  - Isolation of flows highly desirable
  - Bound on packet delay for any source behavior

Comparison with WFQ

- Smaller worst-case delay if

$$r \leq \frac{C}{\mid N \mid -1}; \qquad \mid N \mid\text{: number of flows}$$



- Smaller average delay to low throughput flows

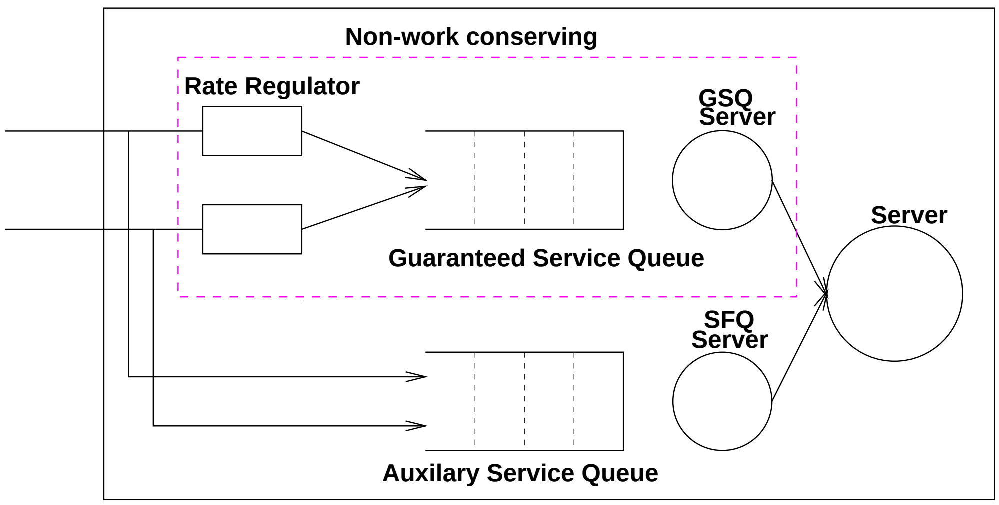
Discussion

- Delay guarantee depends on packet lengths of all flows

$$\beta^j = \sum_{n \in N \wedge n \neq f} \frac{l_n^{max}}{C} + \ldots$$

- Delay guarantee similar to an online scheduler that minimizes unfairness measure
- Providing a priori bounds: estimate β^j
  - Estimate may be large
  - Estimation may be difficult
- Solution: Fair Airport scheduling algorithms

Fair Airport (FA) Algorithms



FA Algorithms: Basic Operation

- Packets join rate controller and Auxiliary Service Queue (ASQ)
- A packet joins Guaranteed Service Queue (GSQ) after leaving rate controller
- GSQ has priority over ASQ
- Packets served from ASQ only when no packets in GSQ
- A packet maybe served from GSQ or ASQ
  - Packet served from GSQ: dequeued from ASQ
  - Packet served from ASQ: dequeued from rate controller

Rate Controller

- Expected arrival time of a packet: the time at which the packet begins service in a server of rate r

$$EAT(p^j) = \max\left\{A(p^j), EAT(p^{j-1}) + \frac{l^{j-1}}{r}\right\}$$

- Eligibility time of p^j: expected arrival time of p^j at rate controller — $EAT^{RC}(p^j)$
  - $EAT^{RC}(p^j)$: computed using only subsequence of packets served from GSQ
  - Example: packets $1, 5$, and 6 served through GSQ and $r = 1pkt/s$

$$EAT^{RC}(p^1) = 0; \quad EAT^{RC}(p^5) = 1; \quad EAT^{RC}(p^6) = 2$$

GSQ and ASQ Schedulers

- GSQ scheduler: any algorithm that guarantees

$$L_{GSQ}(p^j) \leq EAT^{RC}(p^j) + \beta^j$$

- ASQ scheduler: SFQ
  - Packet served from GSQ
    * Packet dequeued from ASQ
    * Start tag of the next packet of flow set to the start tag of packet being removed

Properties of FA

- Fairness guarantee:

$$\left| \frac{W_f(t_1,t_2)}{r_f} - \frac{W_m(t_1,t_2)}{r_m} \right| \leq 2\left(\frac{l_f^{max}}{r_f} + \frac{l_m^{max}}{r_m} \right) + (\beta_f^{max} + \beta_n^{max}) + 2\epsilon$$

  - Holds over variable rate servers

- Deadline guarantees:

$$L(p^j) \leq EAT(p^j) + \beta^j$$

$$L(p^j) \leq \max\{L(p^{j-1}), A(p^j)\} + \frac{l^{max}}{r} + \beta^j + \epsilon$$

Instantiations of FA

- Objective: fair algorithm that achieves separation of rate and delay allocation
  - GSQ scheduler: Delay EDD
  - First fair algorithm that achieves this objective
- Objective: fair algorithm that has delay guarantee of WFQ
  - GSQ scheduler: Virtual Clock

Salient Features

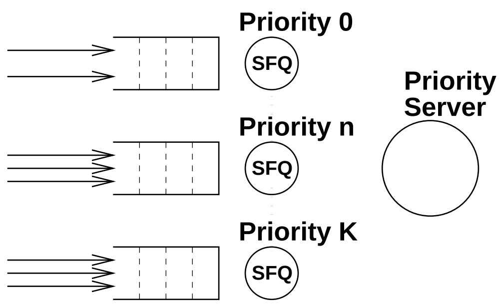
- Fairness over variable rate servers
  - Can be used for achieving hierarchical link sharing
- Different delay guarantees
- Different implementation complexity
  - GSQ: static priority or dynamic priority

SFQ and FA: Tradeoffs

- Computational complexity: lower for SFQ
- Fairness guarantee:
  - Unfairness measure of FA three times that of SFQ
  - SFQ fair regardless of variation in server capacity
  - FA fair only over Fluctuation Constrained servers
- Deadline guarantee:
  - Delay guarantee of FA does not depend on packet lengths of flows
  - Different delay guarantees achievable

Generalization of FA

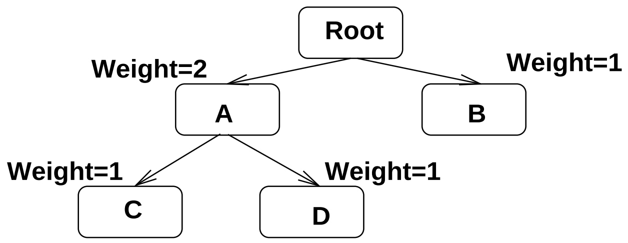
- Prioritized allocation of residual bandwidth



Requirements Revisited

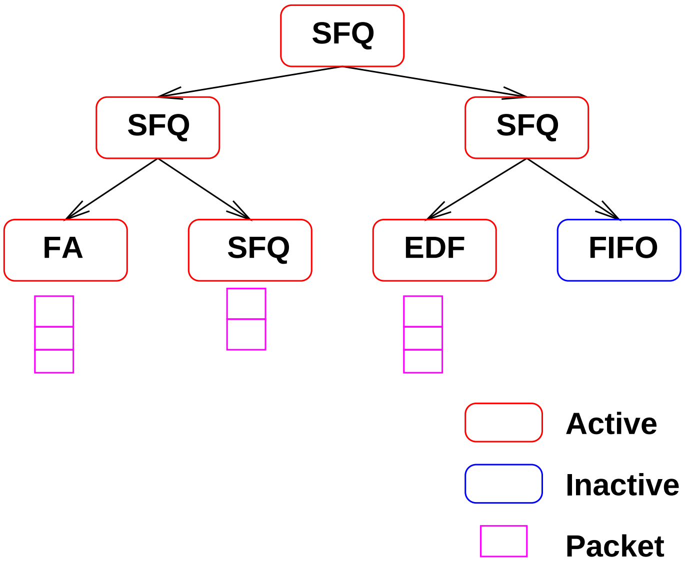
- Fairness for VBR video
- Fairness over variable rate servers for throughput intensive flows
- Low average and worst-case packet delay for low throughput flows
- Hierarchical link bandwidth allocation
- Computationally efficient

Hierarchical Link Bandwidth Allocation



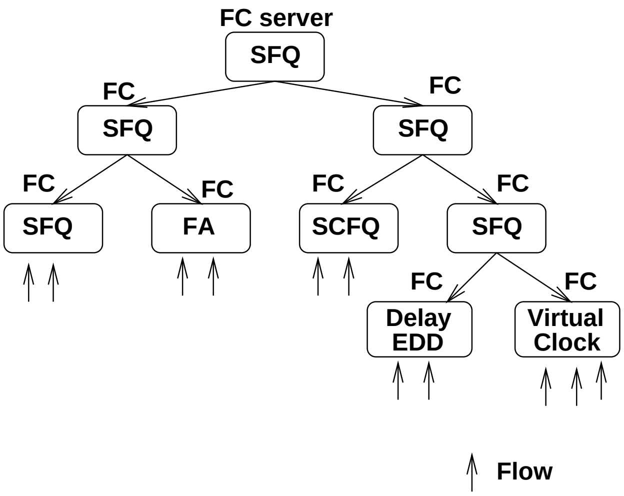
- Can be realized by a hierarchical fair scheduler
- Key requirement: fairness over variable rate servers
- SFQ and FA demonstrated to be fair over variable rate servers

Hierarchical SFQ

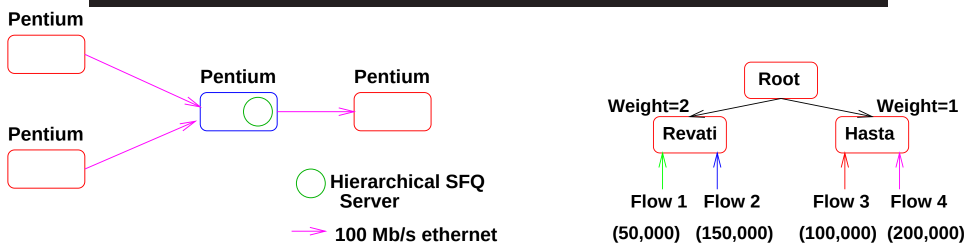


Performance Guarantees

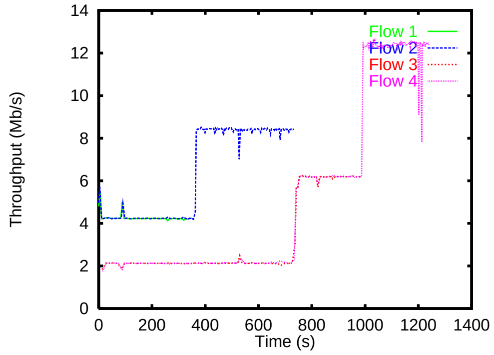
- **Theorem**: If a class is FC server, then each subclass is FC



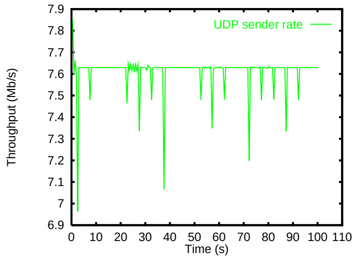
Router Implementation and Evaluation



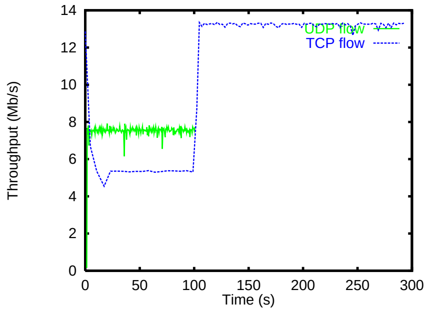
IPv6 Testbed



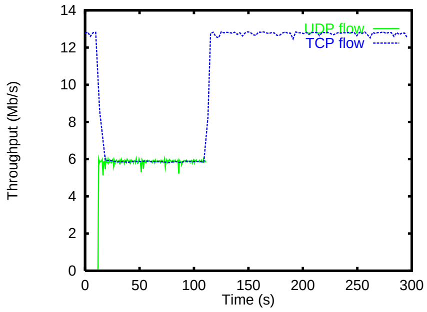
Coexistence of Heterogeneous Control Mechanisms



UDP sender rate



FCFS



Hierarchical SFQ

From a Single Server to Network of Servers

Guaranteed Rate Framework: Motivation

- Many multimedia applications require guaranteed QoS
- Upper bound on end-to-end packet delay depends on:
  - Source traffic characteristics
  - Scheduling algorithm at the switches
- Challenge:
  - Different scheduling algorithms at the switches in heterogeneous networks
  - Heterogeneous source characterization
- Objective: provide bounded packet delay and loss service in heterogeneous networks

GR Scheduling Algorithms

- Guarantee a deadline for a packet based on expected arrival time of the packet - referred to as delay guarantee
- **Definition**: *Guaranteed rate clock* value of packet p_f^j at server i, $GRC^i(p_f^j)$ is:

$$GRC^i(p_f^0) = 0$$

$$GRC^i(p_f^j) = \max\{A^i(p_f^j), GRC^i(p_f^{j-1})\} + \frac{l_f^j}{r_f} \quad j \geq 1$$

p_f^j	:	j^{th} packet of flow f
l_f^j	:	length of p_f^j
$A^i(p_f^j)$	:	the arrival time of packet at server i
r_f	:	the rate associated with flow f

- **Definition**: *GR scheduling algorithm* guarantees that packet p_f^j will be transmitted by $GRC^i(p_f^j) + \beta^i$ where β^i is a constant which depends on the scheduling algorithm and the server

Scheduling Algorithms in GR

Algorithm	β
WFQ, WF^{2}Q, FFQ, WF^{2}Q+	$\frac{l^{max}}{C}$
SCFQ	$\sum_{m \in Q \wedge m \neq f} \frac{l_m^{max}}{C}$
Deficit Round Robbin	$\frac{3F - \phi_f}{C} - \frac{l_f}{r_f}$
Virtual Clock	$\frac{l^{max}}{C}$
Delay EDD, Jitter EDD, RCSP	$d - \frac{l}{r_f}$

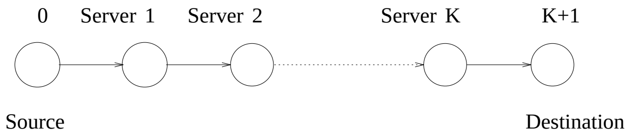
Salient Features of GR

- $GRC^i(p_f^j)$ independent of behavior of other flows
- Isolation of flows highly desirable
  - Sources may misbehave
  - Failure probability of hardware policing devices
- Facilitates end-to-end delay determination

Determining End-to-End Delay Bounds

- Simple approach:
  - Consider each server in isolation
  - Limitations
    * Traffic shape distortion
    * Interdependencies in packet delays at servers
- Solution: analyze a path as a whole assuming that the scheduling algorithm at each server belongs to GR

Methodology



- End-to-end delay of p_f^j:

$$d_f^j \leq GRC^K(p_f^j) + \underbrace{\beta^K + \tau^{K,K+1}}_{\alpha^K} - A^1(p_f^j)$$

$$\Rightarrow d_f^j \leq \left(GRC^K(p_f^j) - A^1(p_f^j)\right) + \alpha^K$$

- Traffic distortion: relationship between $A^K(p_f^j)$ and $A^1(p_f^j)$ not known
- Key observation: $GRC^K(p_f^j)$ depends on $A^K(p_f^j)$, which in turn depends on $GRC^{K-1}(p_f^j) \Rightarrow GRC^K(p_f^j)$ can be related to $GRC^1(p_f^j)$

End-to-End Guarantee

- Two server case:

$$GRC^{i+1}(p_f^j) \leq GRC^i(p_f^j) + \max_{k \in [1..j]} \frac{l_f^k}{r_f} + \alpha^i \quad j \geq 1$$

- Multiple server case:

$$d_f^j \leq \left(GRC^1(p_f^j) - A^1(p_f^j)\right) + (K-1) \max_{n \in [1..j]} \frac{l_f^n}{r_f} + \sum_{n=1}^{n=K} \alpha^n$$

- End-to-end delay completely determined by
  - Computing $GRC^1(p_f^j) - A^1(p_f^j)$
  - Arrival process characteristics at the first server
  - End-to-end delay determination problem reduced to single node problem

End-to-End Delay Bounds

- Delay of packet p_f^j

$$d_f^j = L^K(p_f^j) - A^1(p_f^j) \leq \left(GRC^1(p_f^j) - A^1(p_f^j)\right) + \alpha$$

- $GRC^1(p_f^j) - A^1(p_f^j)$: delay at a single server of rate r_f
  - End-to-end problem reduced to single server problem
- Can be determined for any source behavior

End-to-End Delay Bounds: Example

- Leaky bucket constrained flow:

$$AP_f(t_1, t_2) \leq \sigma + \rho(t_2 - t_1)$$

σ	:	burstiness
ρ	:	average rate

$$GRC^1(p_f^j) - A^1(p_f^j) \leq \frac{\sigma}{\rho}$$

For PGPS:

$$d_f^j \leq \frac{\sigma}{\rho} + \frac{(K-1)\max_{n\in[1..j]} l_f^n}{\rho} + \sum_{n=1}^{n=K} \beta^n$$

- Probabilistic bounds can be derived

Salient Features

- Interpret $GRC^1(p_f^j) - A^1(p_f^j)$ as queuing delay at a server of capacity r_f
- Source can determine end-to-end delay without specifying the shape of the traffic to the network
- Delay bounds are tight