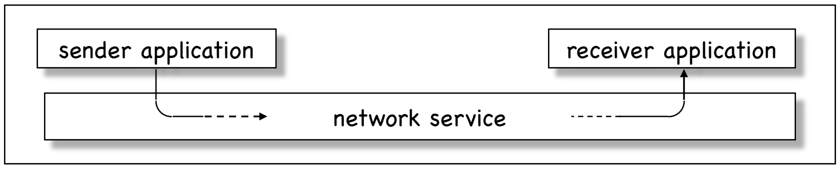
Real-Time Communication

- Integrated Services: Integration of variety of services with different requirements (real-time and non-real-time)
- Traffic (workload) characterization
- Scheduling mechanisms
- Admission control / Access control (policing)
- Deterministic *vs.* stochastic analysis
  - Traffic characterization
  - Performance guarantees
- Later we will
  - talk about real-time communication management in the real-world (intserv, some diffserv)
  - wrap up things with a theoretical framework (network calculus)

Providing Real-Time Guarantees



Traffic Specification

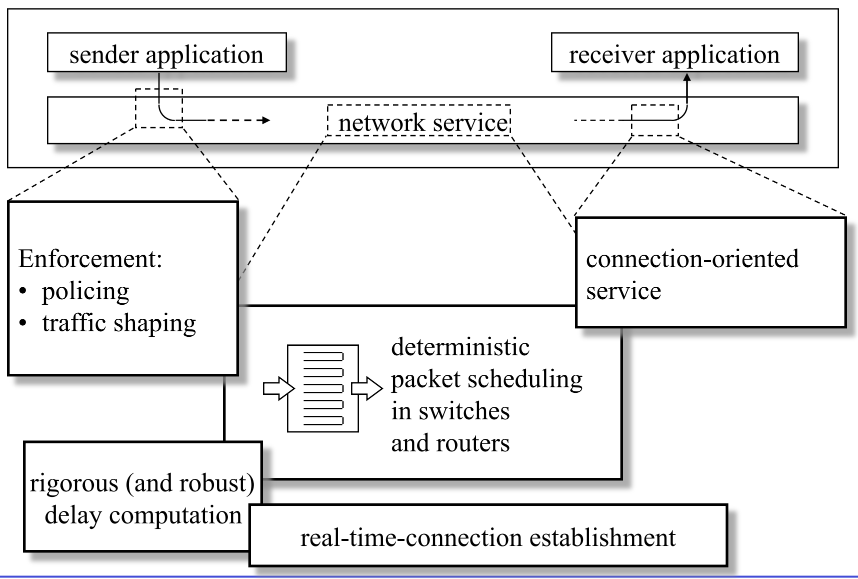
- **packet sizes**
- **packet inter-arrival times**
- **general traffic descriptors**

Performance Requirements

- **delay**
- jitter
- **bandwidth**
- packet loss

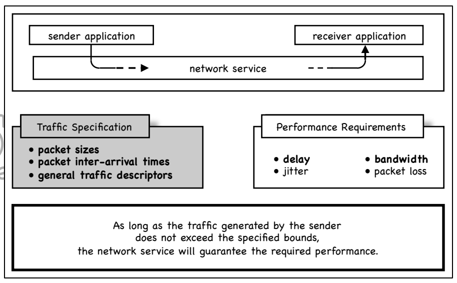
As long as the traffic generated by the sender
does not exceed the specified bounds,
the network service will guarantee the required performance.

Real-Time Guarantees: Mechanisms

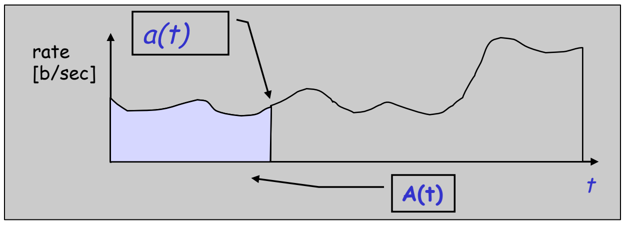

sender application
receiver application
network service
Enforcement:
• policing
• traffic shaping
connection-oriented service
deterministic packet scheduling in switches and routers
rigorous (and robust) delay computation
real-time-connection establishment

Real-Time Communication

- Integrated Services: Integration of variety of services with different requirements (real-time and non-real-time)
- Traffic (workload) characterization
- Scheduling mechanisms
- Admission control / Access control (policing)
- Deterministic *vs.* stochastic analysis
  - Traffic characterization
  - Performance guarantees
- Later we will
  - talk about real-time commun
    world (intserv, some diffserv)
  - wrap up things with a theore


sender application
receiver application
network service
Traffic Specification
• packet sizes
• packet inter-arrival times
• general traffic descriptors
Performance Requirements
• delay
• jitter
• bandwidth
• packet loss
As long as the traffic generated by the sender does not exceed the specified bounds, the network service will guarantee the required performance.

Traffic Description: Traffic Bounding Functions



Arrival as stochastic process
$A = \{A(t),\ t \geq 0\}$

A and *a* are poor traffic descriptors:
– time dependent

Deterministic traffic arrival descriptors (time-**in**dependent)

Maximum Traffic Function	$b(I) \geq \max_{t>0}\{A(t+I) - A(t)\}$
Maximum Rate Function	$b(I)/I \geq \max_{t>0}\{A(t+I) - A(t)\}/I$

© R. Bettati

Traffic Bounding Function *b(.)*

Let *b(.)* be a monotonically increasing function.

We call *b(.)* a **deterministic traffic constraint function** of a connection if during any interval of length *I*, the number of bits arriving during the interval is no greater than *b(I)*.

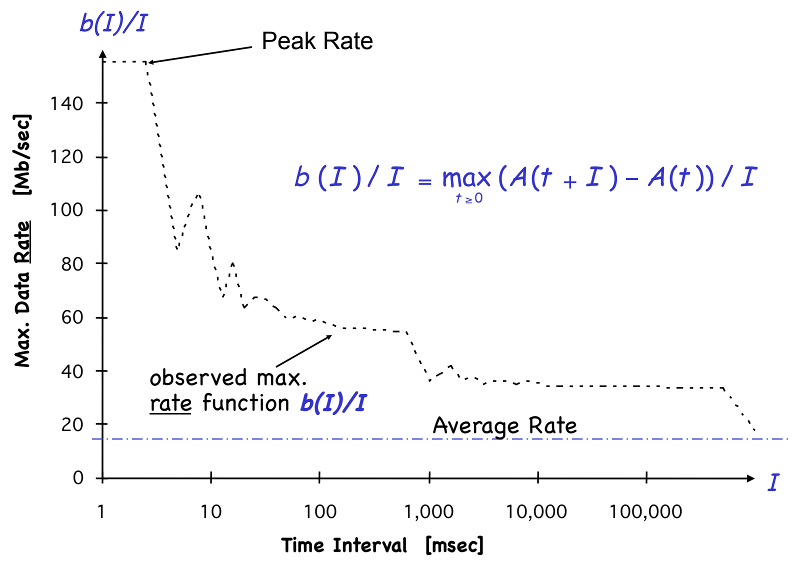
Let *A[t1,t2]* be the number of packets arriving during interval *[t1,t2]*. Then, *b(.)* is a traffic constraint function if

$$\forall s, I > 0, \quad A[s, s+I] \leq b(I)$$

- Traffic models inherently define a traffic constraint function.
- The accuracy of models can be compared by comparing their constraint functions.

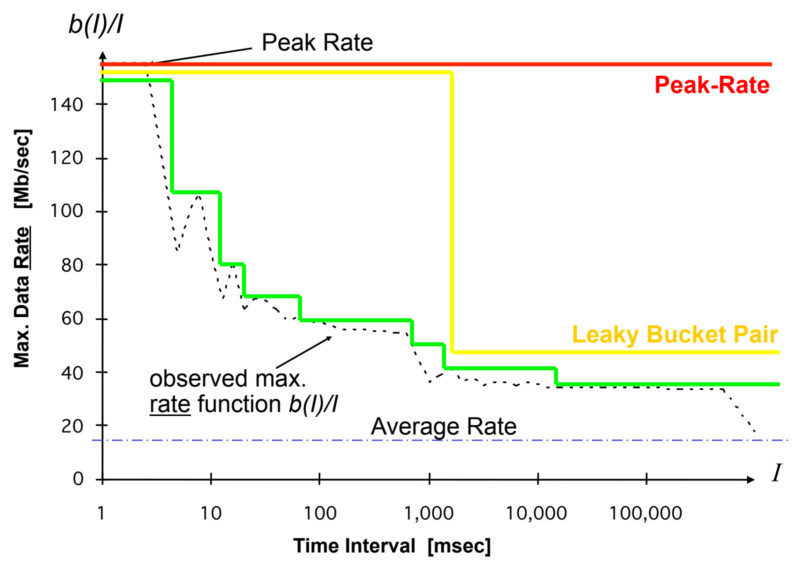
© R. Bettati

Maximum Traffic Functions



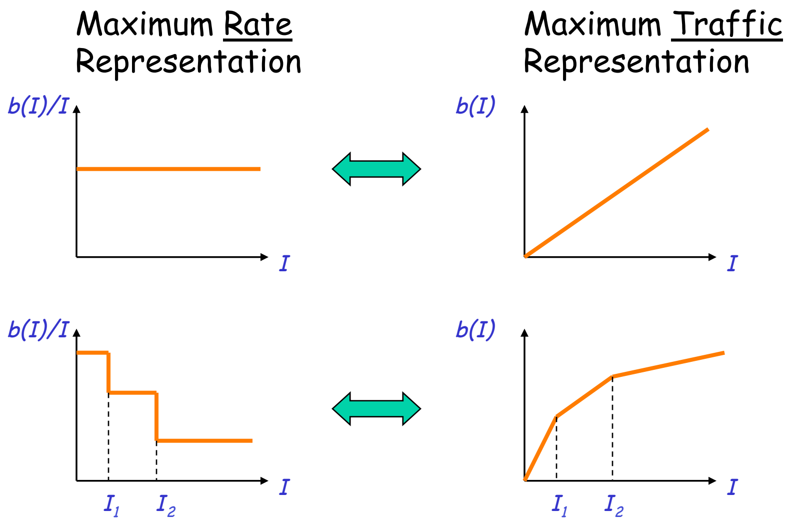
© R. Bettati

Traffic Bounding Functions



© R. Bettati

Maximum Rate *vs.* Maximum Traffic Functions



© R. Bettati

Traffic Models

Deterministic Models:

1. Periodic model: *(e, p)*
2. Deferred Server, Sporadic Server model: (e_S, p_S)
3. (σ, ρ) model [Cruz]
4. Leaky bucket model [Turner, ...]: (β, ρ)
5. $(x_{min}, x_{ave}, I, s_{max})$ model [Ferrari & Verma]
6. D-BIND model (Deterministic Bounding Interval Length Dependent) [Knightly & Zhang]
7. Γ-functions [Zhao]

Probabilistic Models:

1. S-BIND model (Stochastic Bounding Interval) [Knightly]
2. Markov-Modulated Poisson Processes

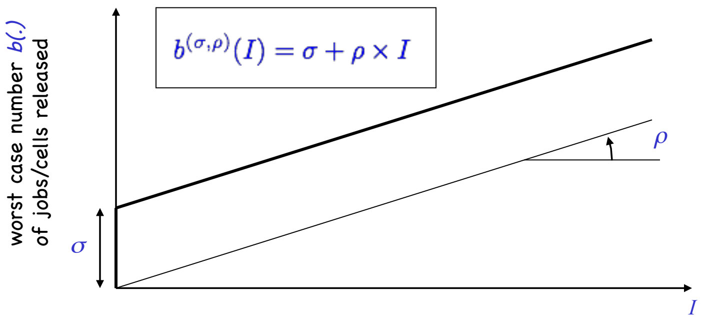
© R. Bettati

Cruz' (σ, ρ) Model

"If the traffic is fed to a server that works at rate ρ while there is work to be done, the size of the backlog will never be larger than σ."

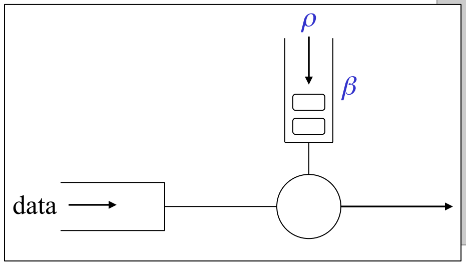
IOW:
The number of jobs/cells released during any interval I does not exceed $\rho I+\sigma$.

Graphical representation:

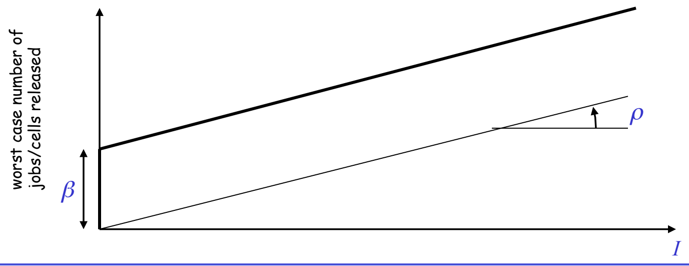


© R. Bettati

The Leaky Bucket Model



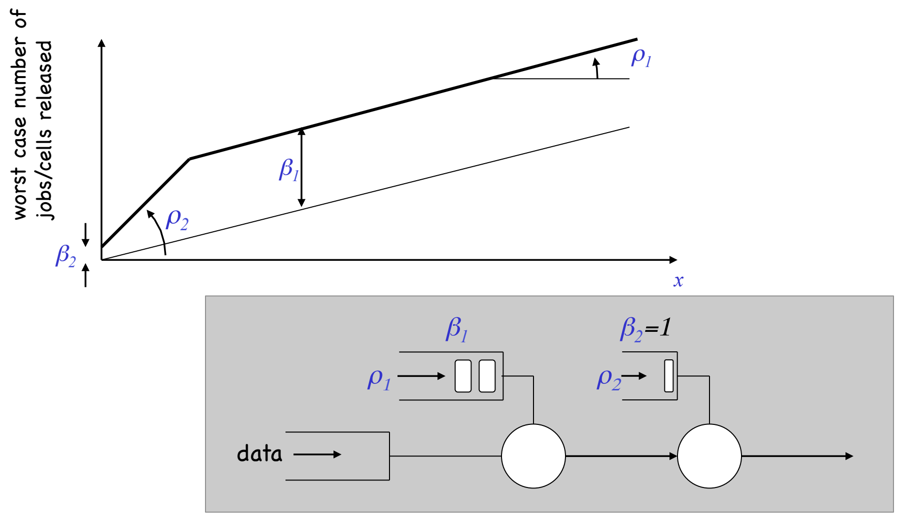
- Implementation:
  - Maintain counter for each traffic stream.
  - Increment counter at rate ρ, to maximum of β.
  - Each time a packet is offered, the counter is checked to be > 0.
  - If so, decrement counter and forward packet; otherwise drop packet.



© R. Bettati

Concatenating Leaky Buckets

Q: What about limiting the maximum cell/frame rate?

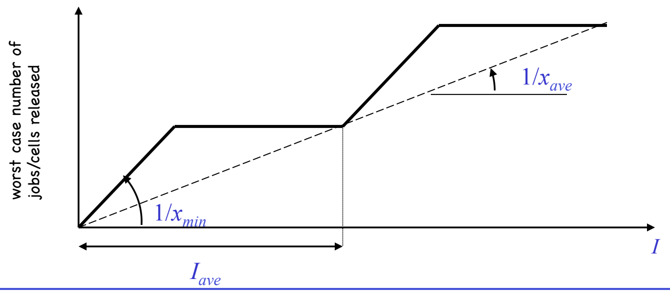


© R. Bettati

$(x_{min}, x_{ave}, I_{ave}, s_{max})$ model [Ferrari & Verma]

- x_{min} : minimum packet interarrival time
- x_{ave} : average packet interarrival time
- I_{ave} : averaging interval length
- s_{max} : maximum packet length

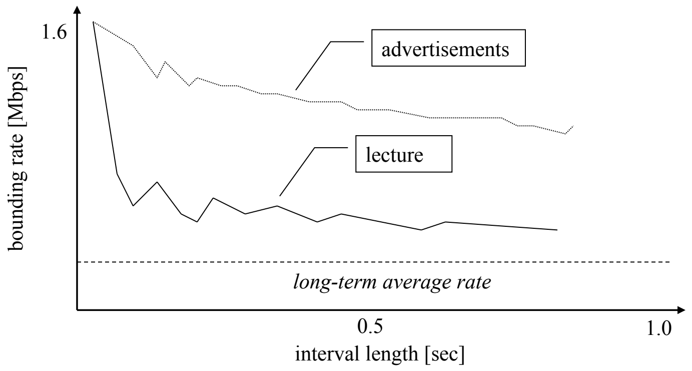
$$b(x_{\min}, x_{ave}, I_{ave}, s_{\max})(I) = \left(\min\left(\left\lceil \frac{t \bmod I_{ave}}{x_{\min}} \right\rceil, \left\lceil \frac{I_{ave}}{x_{ave}} \right\rceil \right) + \frac{I}{I_{ave}} \right) s_{\max}$$



© R. Bettati

D-BIND [Knightly & Zhang]

- Other models do not accurately describe burstiness.
- Rate-interval representation:



- Model traffic by multiple rate-interval pairs: (R_k, I_k), where rate R_k is the worst-case rate over every interval of length I_k.

© R. Bettati

D-BIND (2)

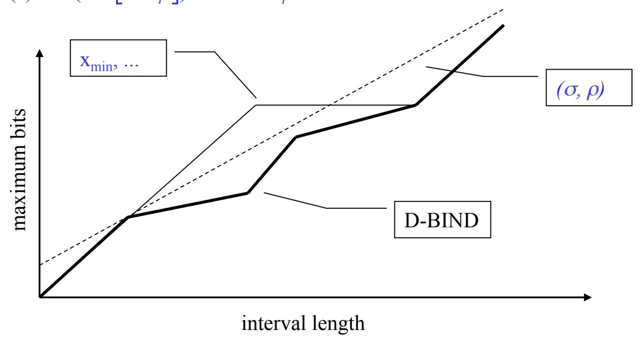
- Constraint function for D-BIND model with P rate-interval pairs:

$$b(t) = \frac{R_k I_k - R_{k-1} I_{k-1}}{I_k - I_{k-1}}(t - I_k) + R_k I_k, \qquad I_{k-1} \le t \le I_k$$

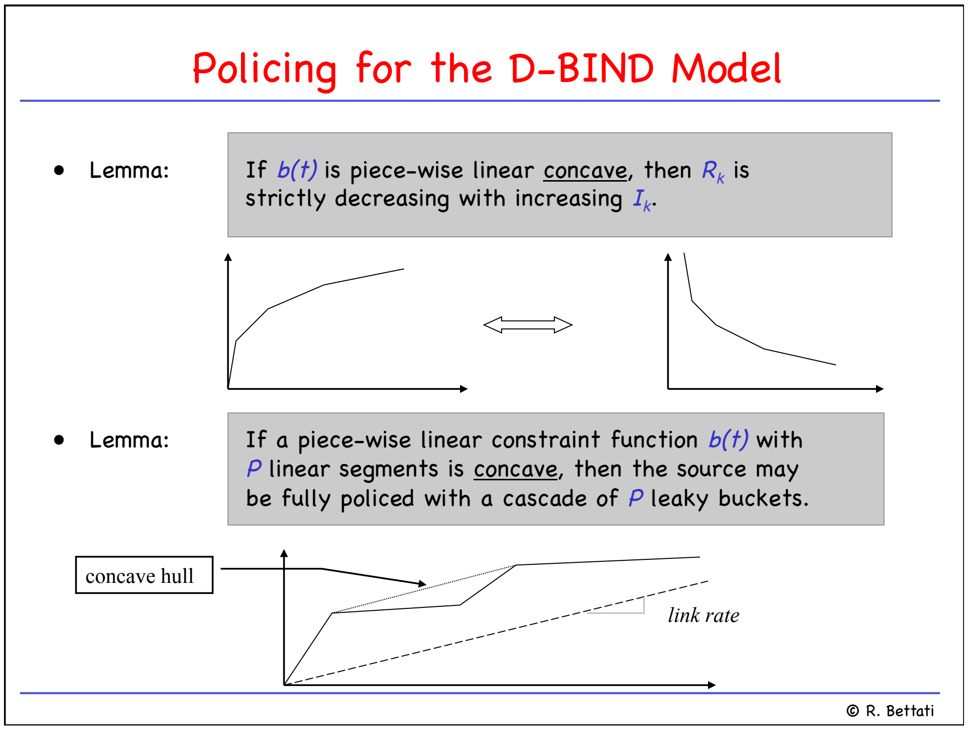
$$b(0) = 0$$

$$b(t) = b(t - \lfloor t / I_P \rfloor) \text{ for } t > I_P$$

- Comparison:

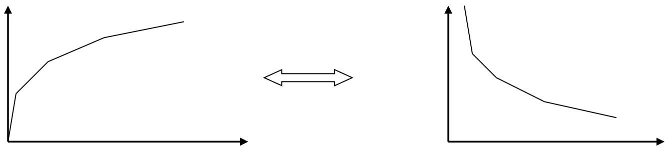


© R. Bettati

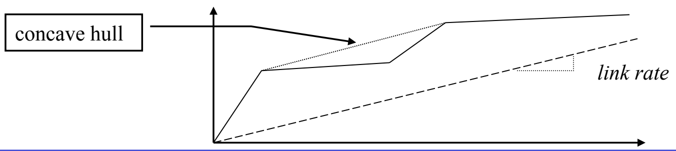


Policing for the D-BIND Model

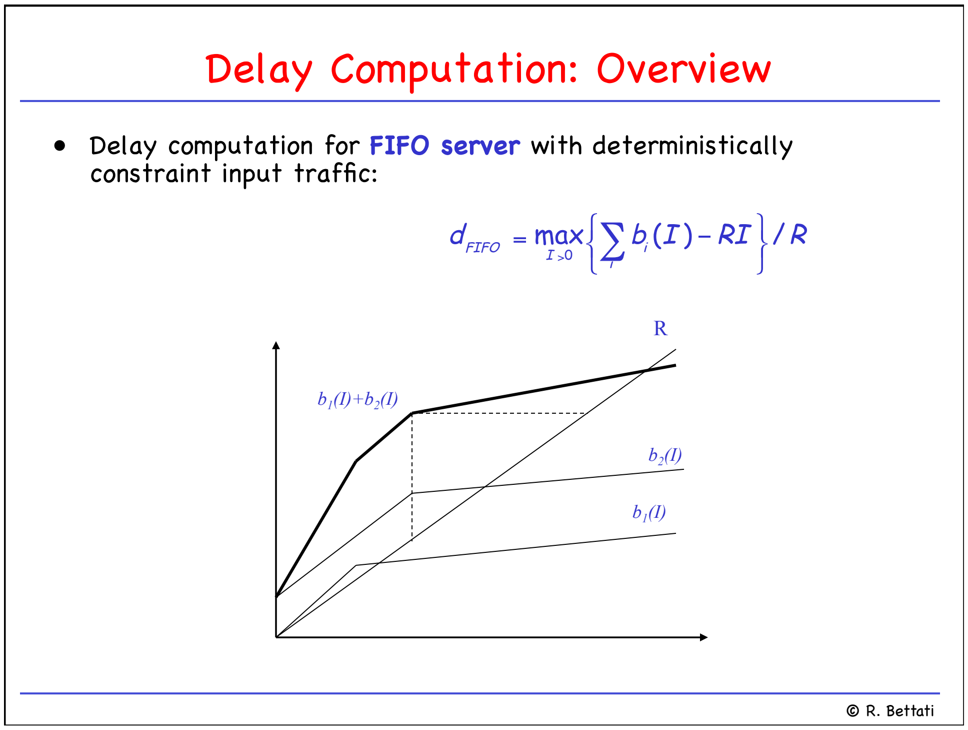
- Lemma: If $b(t)$ is piece-wise linear concave, then R_k is strictly decreasing with increasing I_k.



- Lemma: If a piece-wise linear constraint function $b(t)$ with P linear segments is concave, then the source may be fully policed with a cascade of P leaky buckets.



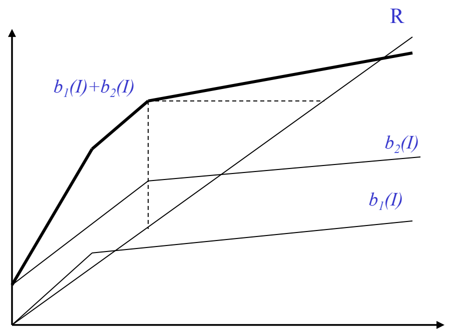
© R. Bettati



Delay Computation: Overview

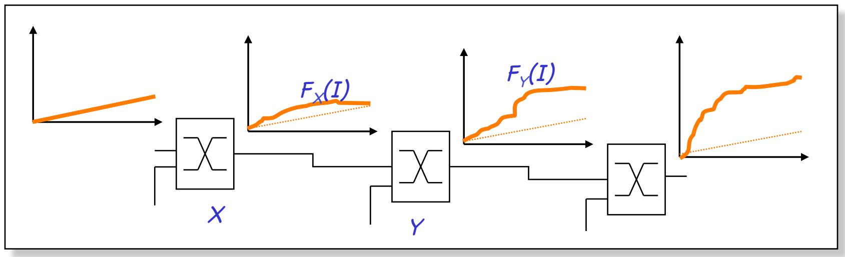
- Delay computation for **FIFO server** with deterministically constraint input traffic:

$$d_{FIFO} = \max_{I>0}\left\{\sum_i b_i(I) - RI\right\}/R$$



© R. Bettati

End-to-End Analysis



- Traffic regulation: reshape traffic to adhere to traffic function.
- Alternative: re-characterize by accounting for burstiness added by queueing delays

$$F_Y(I) = F_X(I+d_y)$$

  - where d_Y is delay on Server Y.
- Deterministic Case:

$$d^{e2e} = \sum_{X \in R} d_X$$

Real-Time Communication

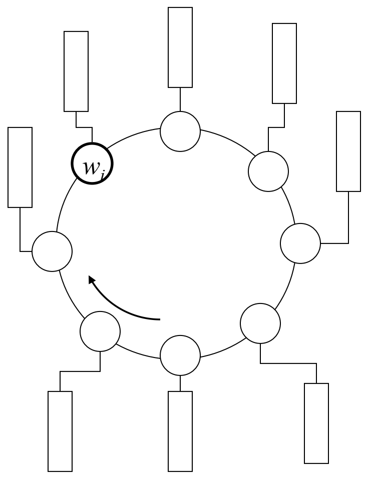
- Integrated Services: Integration of variety of services with different requirements (real-time <u>and</u> non-real-time)
- Traffic (workload) characterization
- Scheduling mechanisms
- Admission control / Access control (policing)
- Deterministic *vs.* stochastic analysis
  - Traffic characterization
  - Performance guarantees
- Later we will
  - talk about real-time communication management in the real-world (intserv, some diffserv)
  - wrap up things with a theoretical framework (network calculus)

Switch Scheduling

- Work-conserving (greedy) *vs.* non-work-conserving (non-greedy) mechanisms.
- Rate-allocating disciplines: Allow packets to be served at higher rates than the guaranteed rate.
- Rate-controlled disciplines: Ensures each connection the guaranteed rate, but does not allow packets to be served above guaranteed rate.

- Priority-based scheduling:
  - fair queuing
  - virtual clock
  - earliest due date (EDD)
  - rate-controlled static priority (RCSP)
- Weighted Round-Robin scheduling:
  - WRR

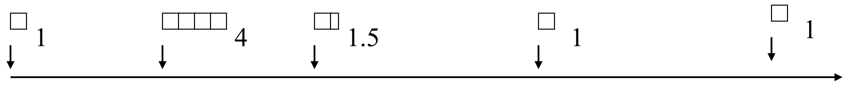
Bit-by-Bit Weighted Round-Robin



- bit-by-bit round robin
- each connection is given a weight
- each queue served in FIFO order

Fair Queueing [Demers, Keshav, Shenker]

- Emulate Bit-by-Bit Round Robin by prioritizing packets.
- Prioritize packets on basis of their finish time f_j:
  - a_j: arrival time of j-th packet
  - e_j: length of packet
  - f_j: finish time
  - BW: allocated fraction of link bandwidth
- Example: $f_j = \max(f_{j-1}, a_j) + e_j / BW$



- Complications:
  - What if connections dynamically change?

© R. Bettati

Guaranteed-Rate Scheduling Alg's [Goyal, Lam, Vin]

- Guaranteed-Rate Clock *GRC* associated with each packet. (intuitively, expected arrival time)
- Flow *f* is associated rate r_f (bits/sec)
- Let p^j_f and l^j_f denote jth packet of flow f and its length.
- $CRC^i(p^j_f)$: guaranteed-rate clock value of packet p^j_f
- $A^i(p^j_f)$: arrival time of packet p^j_f

$$GRC^i(p^0_f) = 0$$

$$GRC^i(p^j_f) = max\{A^i(p^j_f), GRC^i(p^{j-1}_f)\} + \frac{l^j_f}{r_f} \quad j \geq 1$$

- Examples are
  - Virtual-Clock
  - Packet-by-Packet Generalized Processor Sharing
  - Self-Clocked Fair Queueing
  - etc.

© R. Bettati

Virtual Clock Algorithm [L.Zhang]

- Emulate time-division multiplex (TDM) mechanism
  - Pre-allocation of slots eliminates interference among users.
- However:
  - TDM: when some connections idle, the slots assigned are idle
  - VC: idle slots are deleted from TDM frames

- auxiliary virtual clock (*auxVC$_j$*): finish time of *j*-th packet.
- virtual tick (*Vtick$_j$*) :time to complete transmission of ready *j*-th packet.

  $Vtick_j = e_j/BW$

- Replace f_j by $Vtick_j$: VC becomes identical to WFQ algorithm!

- Will look at delay analysis later.

Virtual Clock: Vanilla Implementation

1. Upon flow set-up of F_i with average transmission rate AR_i: compute value $Vtick_i = 1/AR_i$ (if variable-size packets, scale Vtick appropriately)
2. Upon arrival of first packet from F_i: VC_i := real_time
3. Upon receiving packet j from F_i:
   - $VC_i := VC_i + Vtick_i$
   - timestamp packet j with value VC_i
4. Transmit packets in order of increasing VC's
5. When switch runs out of buffer space, drop the last packet from the queue.

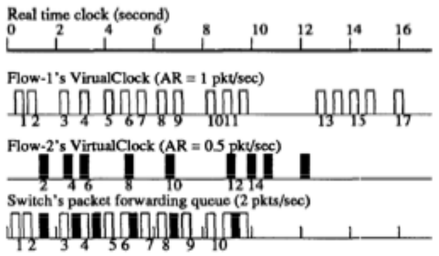
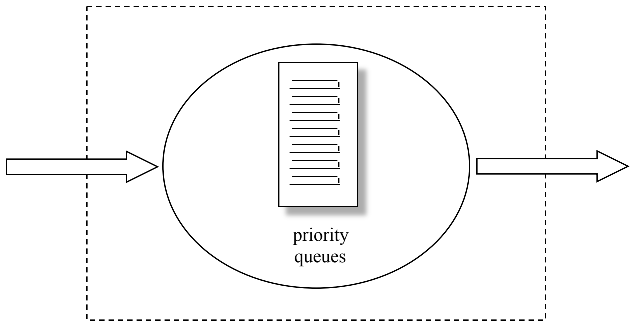
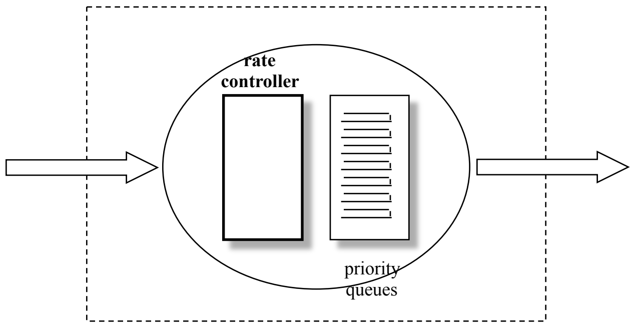

Real time clock (second)
0
2
4
6
8
10
12
14
16
Flow-1's VirualClock (AR = 1 pkt/sec)
1 2 3 4 5 6 7 8 9 10 11 13 15 17
Flow-2's VirtualClock (AR = 0.5 pkt/sec)
2 4 6 8 10 12 14
Switch's packet forwarding queue (2 pkts/sec)
1 2 3 4 5 6 7 8 9 10

Figure 1: Real time, Virtual Clock, and packet processing order.

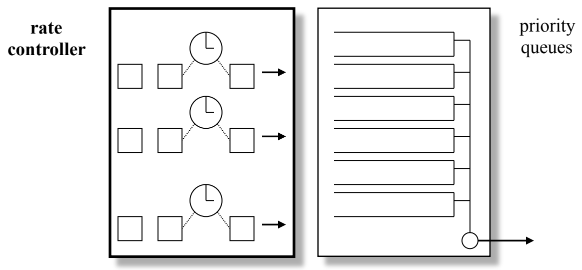
Rate-Controlled Static Priority (RCSP) [Zhang&Ferrari]



RCSP (2)



Traffic Regulation in RCSP



- Hold packets in regulator to guarantee minimum inter-packet arrival time.

  $r_{i,j} = max(a_{i,j}, r_{i,j-1}+p_i)$
- Implementation: buffer and timers in traffic regulator.
- Buffer requirements:

$$B_{ik} = \left(\left\lceil \frac{d_{ik}}{p_i} \right\rceil + \left\lceil \frac{d_{ik-1}}{p_i} \right\rceil\right) e_i$$

© R. Bettati

Is it Necessary to Regulate?

- [Liebeherr, Wrege, Ferrari, Transactions on Networking, 1995]
- Generalization of schedulability for arbitrary traffic constraint functions $b(I)$:

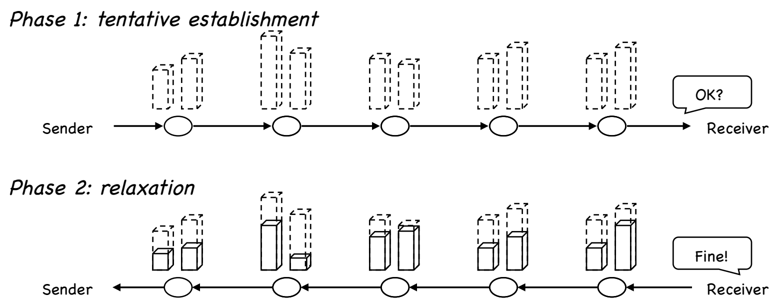
Theorem: A set N of connections that is given by $\{b_j, d_j\}$ is schedulable according to a static-priority algorithm if and only if for all priorities p, and for all $I >= 0$ there is a t with $t <= d_p - s_p^{min}$ such that:

$$\forall I, \exists t \leq d_p - s_p^{\min} : I + t \geq \sum_{j \in C_p} b_j(I) - s_p^{\min} + \sum_{q=1}^{p-1} \sum_{j \in C_q} b_j((I+t)^-) + \max_{r>p}\{s_r^{\max}\}$$

© R. Bettati

Earliest Due Date (EDD) [Ferrari]

- based on EDF
- delay-EDD *vs.* jitter-EDD
- works for periodic message models (single packet in period): $(p_i, 1, D_i)$
- partition end-to-end deadline D_i into local deadlines $D_{i,k}$ during connection establishment procedure.
- 2-Phase establishment procedure:



© R. Bettati

Delay EDD

- Upon arrival of Packet j of Connection i:
  - Determine effective arrival time: $a^e_{i,j} = max(a^e_{i,j-1} + p_i, a_{i,j})$
  - Stamp packet with local deadline: $d_{i,j} = a^e_{i,j} + D_{i,k}$
  - Process packets in EDF order.
- Delay EDD is greedy.
- Can be mapped into special case of Sporadic Server.
- Acceptance test (Δ = total density): $\Delta + 1/p_i < 1 - 1/p_{min}$
- Offered local deadline: $LD_i = \min(p_i, 1/(1-\Delta-1/p_{min}))$
- Problem with EDD: **jitter**
  - **max** end-to-end delay over k switches: $\sum_k D_{i,k}$
  - **min** end-to-end delay over k switches: k

© R. Bettati

Jitter EDD

- Problem with Delay-EDD: does not control jitter. This has effect on buffer requirements.
- Jitter-EDD maintains **Ahead Time** $ah_{i,j}$, which is the difference between local relative deadline $D_{i,k-1}$ and actual delay at Switch $k-1$.
- Ahead time is stored in packet header (alternatively, we use global time synchronization)
- Upon receiving the j-th packet of Connection i with $ah_{i,j}$ at time $a_{i,j}$:
  - Calculate ready time as Switch k:

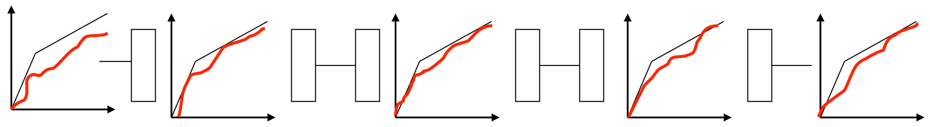
    $a^e_{i,j}=\max(a^e_{i,j-1} + p_i , a_{i,j})$

    $r_{i,j} = \max(a^e_{i,j} , a_{i,j} + ah_{i,j})$
  - Stamp packet with deadline $d_{i,j}=r_{i,j}+D_{i,k}$ and process according to EDF starting from ready time $r_{i,j}$.
- Result: Regenerate traffic at each switch.

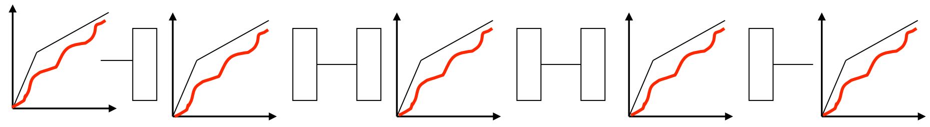
© R. Bettati

Rate Control *vs.* Jitter Control

- Rate Control



- Jitter Control



© R. Bettati

Simple EDF with Arbitrary Arrival Functions

[Liebeherr, Wrege, Ferrari: Transactions on Networking, 1995]

Theorem: A set Π of connections that is given by $\{b_i; d_i\}_{i \in \Pi}$ and $d_i \leq d_j$ whenever $i<j$ is EDF schedulable if and only if for all $I \geq d_1$:

$$I \geq \sum_{j\in\Pi} b_j(I-d_j) + \max_{k,d_k>I} \{s_k^{max}\}$$

where

$$\max_{k,d_k>I} \{s_k^{max}\} = 0 \quad for \quad I > \max_{k\in\Pi}\{d_k\}$$

Informal "proof": A deadline violation occurs at time I if the maximum traffic arrivals with deadline before or at time I exceeds I, i.e.:

$$I < \sum_{j\in\Pi} b_j(I-d_j)$$

© R. Bettati

EDF Test for Special Cases: Example (σ,ρ)

- For some traffic models, closed-form expressions for the schedulability test exist.
- For (σ, ρ) traffic:

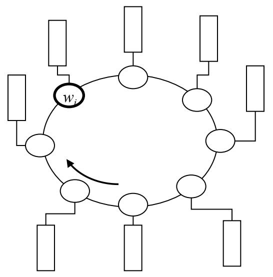
$$\begin{cases} I \geq \sum_{i=1}^{j} \sigma_i + \rho_i(I-d_i) + \max_{k>j}\{s_k^{max}\} \quad for \ \ d_j \leq I < d_{j+1} : 1 \leq j < |\Pi| \\ I \geq \sum_{i=1}^{|\Pi|} \sigma_i + \rho_i(I-d_i) \quad for \ \ I > d_{|\Pi|} \end{cases}$$

- A closed form for the delay can be given as follows:

$$d_j = \frac{\sigma_j + \sum_{i=1}^{j-1}(\sigma_i - \rho_i d_i) + \max_{k>j}\{s_k^{max}\}}{1-\sum_{i=1}^{j-1}\rho_i}$$

© R. Bettati

Weighted Round Robin (WRR)



- Each connection *i* is assigned a weight w_i, i.e., it is allocated w_i slots during each round.
- Slot: time to transmit maximum-sized packet.

- Traffic model:
  - periodic (p_i, e_i, D_i)
  - variable bit rate models possible
- Realizations:
  - greedy WRR
  - Stop-and-Go (SG)
  - Hierarchical Round Robin (HRR)

Throughput and Delay Guarantees

- Each connection *i* is guaranteed w_i slots in each rounds.
- Round length *RL* : upper bound on sum of weights (design parameter)

$$\sum w_i \leq RL$$

- Constraints:

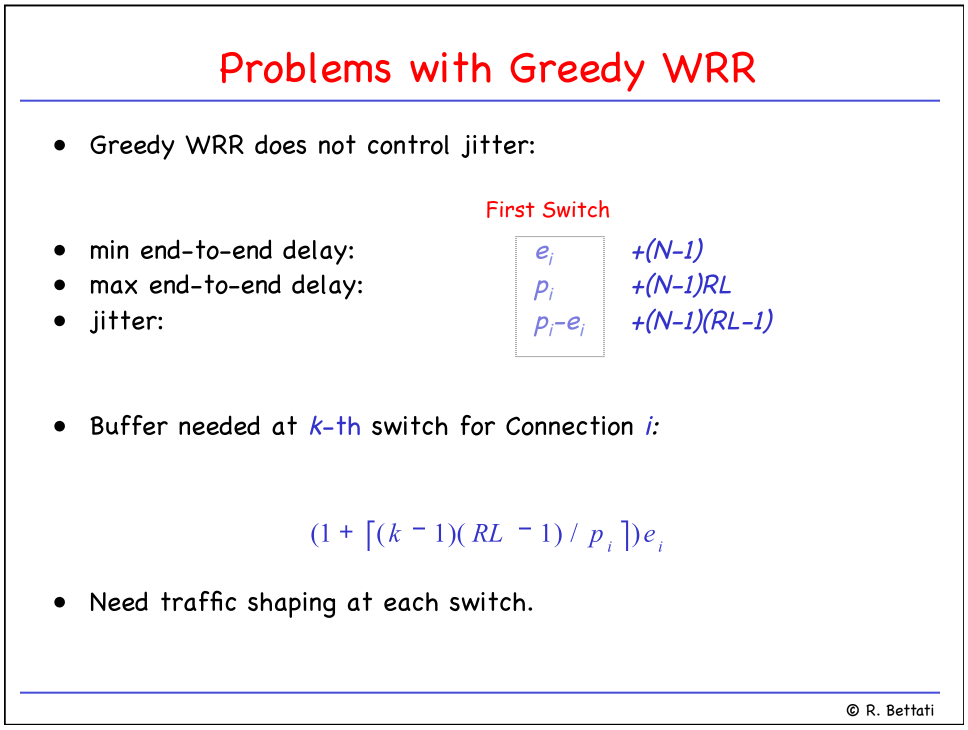
  1. $RL \leq p_{\min}$

  2. $w_i \geq \left\lceil \frac{e_i}{\lfloor p_i / RL \rfloor} \right\rceil$

- Delays:
  - at first switch: $\left\lceil \frac{e_i}{w_i} \right\rceil RL$
  - downstream: once packet passes first switch, it is immediately eligible on switches downstream -> has to wait at most *RL*

    => end-to-end delay through *N* switches:

$$W_i \leq \left(\lceil e_i / w_i \rceil + N - 1 \right) RL \leq p_i + (N-1) RL$$



Problems with Greedy WRR

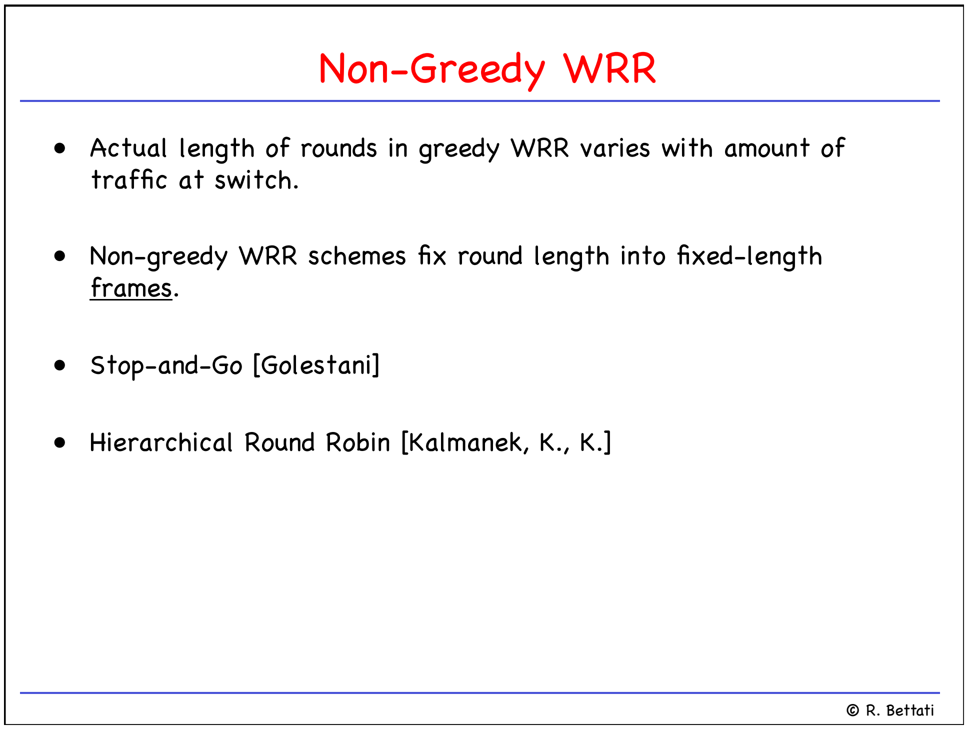
- Greedy WRR does not control jitter:

	First Switch	
min end-to-end delay:	e_i	$+(N-1)$
max end-to-end delay:	p_i	$+(N-1)RL$
jitter:	p_i-e_i	$+(N-1)(RL-1)$

- Buffer needed at *k-th* switch for Connection *i:*

$$(1 + \lceil (k-1)(RL-1)/p_i \rceil)e_i$$

- Need traffic shaping at each switch.

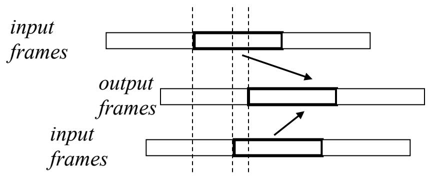


Non-Greedy WRR

- Actual length of rounds in greedy WRR varies with amount of traffic at switch.
- Non-greedy WRR schemes fix round length into fixed-length frames.
- Stop-and-Go [Golestani]
- Hierarchical Round Robin [Kalmanek, K., K.]

Stop & Go [Golestani, 1990]

- Frame-based: divide time in **frames** of length *RL*.
- Packet arriving during frame at input link is eligible for transmission during **next** frame on output link.



- Stop-and-Go is *not* work-conserving.
- Traffic model **[*(r, RL)* smooth traffic]**: during each frame of length *RL*, the total number of bits transmitted by source does not exceed *rRL* bits.

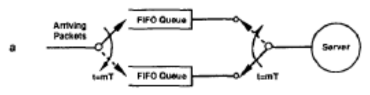
- **Proposition:** If the connection satisfies (r,RL) smoothness at the input of the first server, and each server ensures that packets will always go out on the next departing frame, the connection will satisfy (r,RL) smoothness at each server throughout the network.

© R. Bettati

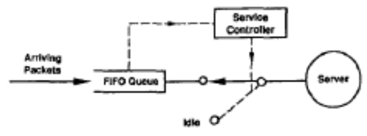
Stop & Go: Implementation

- Implementation of scheduler is not defined by Stop-and-Go frameworks.

- Implementation 1: FIFO scheduler with double-queue structure



- Implementation 2:

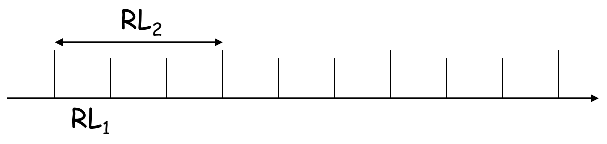


© R. Bettati

Multi-Frame Stop-and-Go

[For example, Zhang&Knightly: "Comparison of RCSP and SG", ACM Multimedia, 4(6) 1996]

- Problem with Stop-and-Go (or any other frame-based approach): **delay-bandwidth coupling**
  - Delay of packet is bounded by a multiple of frame time. This is a problem, for example for low-bandwidth, low-delay connections. (Why?)
- Solution: Use multi-level framing. Example:



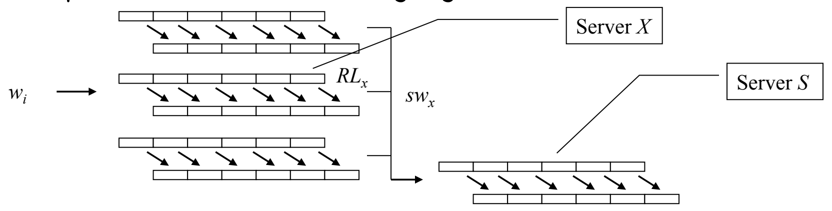
- Hierarchical framing with n levels with frame sizes $RL_1, \ldots, RL_n$, where $RL_{m+1}=K_m RL_m$ for $m = 1, \ldots, n-1$.
- Stop-and-Go rule for packets of level-p connection: Packets that arrived during a RL_p frame will not become eligible until the start of the next RL_p frame.
- Packets with smaller frame size have higher priority (non-preemptively) over packets with larger frame size.

© R. Bettati

Hierarchical Round Robin

[Kalmanek, Kanadia, Keshav, 1990]

- End-to-end delay and jitter of S&G depends on RL only.
- How about having multiple S&G servers, with different RL's, and multiplex them on the same outgoing link?



- Server X is seen as periodic stream of requests by Server S, with
  - $e_x = sw_x$, $p_x = RL_x$, $D_x = RL_x$
  - schedule using rate-monotonic scheduler
  - Configuration time test: check whether task set $\{(sw_x, RL_x, RL_x)\}$ is schedulable.
- Admission Control Test:

– Bandwidth test:	check sum of required w_i's <= sw_x
– Delay test:	End-to-end delay: $p_i + N\ RL_x$
– Jitter test:	$2\ RL_x$, with buffer requirement $2\ w_i$

© R. Bettati