

# Reti di Calcolatori

## Tecniche per il controllo del traffico

Giorgio Ventre

Gruppo di Ricerca sull'Informatica Distribuita  
Dipartimento di Informatica e Sistemistica  
Università di Napoli Federico II

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Nota di Copyright

Quest'insieme di trasparenze è stato ideato e realizzato dai ricercatori del Gruppo di Ricerca sull'Informatica Distribuita del Dipartimento di Informatica e Sistemistica dell'Università di Napoli e del Laboratorio Nazionale per la Informatica e la Telematica Multimediali. Esse possono essere impiegate liberamente per fini didattici esclusivamente senza fini di lucro, a meno di un esplicito consenso scritto degli Autori. Nell'uso dovrà essere esplicitamente riportata la fonte e gli Autori. Gli Autori non sono responsabili per eventuali imprecisioni contenute in tali trasparenze né per eventuali problemi, danni o malfunzionamenti derivanti dal loro uso o applicazione.

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

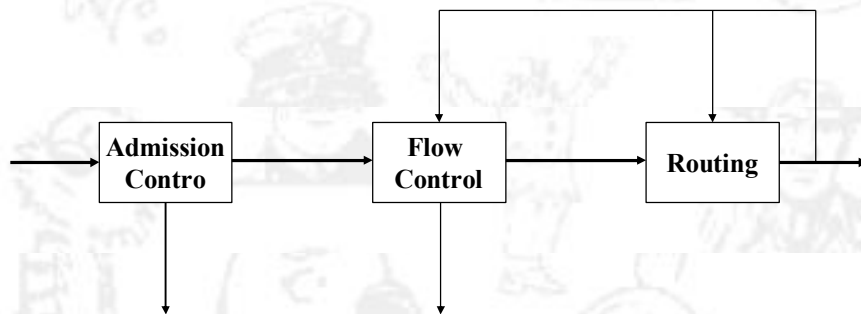
## Il problema del Controllo

- | A prescindere dalle modalità di trasporto e commutazione, una rete è un sistema che deve essere controllato e gestito
  - » Miglioramento QoS
  - » Ottimizzazione uso risorse
  - » Aumento redditività
- | I meccanismi di controllo disponibili sono molteplici
  - » Routing
  - » Admission Control
  - » Flow Control

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Il problema del Controllo



Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## II Flow Control

- | In una rete ideale
  - »  $T_{\text{offered}} = T_{\text{throughput}} = T_{\text{received}}$
- | In generale, purtroppo
  - »  $T_{\text{offered}} > T_{\text{throughput}} > T_{\text{received}}$
- | Il Flow Control disciplina la trasmissione in modo da evitare problemi legati alla limitata capacità
  - » Buffer Overflow nei nodi di rete
  - » Buffer Overflow nei terminali
- | Ma dove esercitare il controllo?

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## II Flow Control

- | Esempio: controllo del traffico su una autostrada
- | Controllo lungo il percorso
  - » Limiti di velocità
- | Controllo agli accessi
  - » Numero dei varchi di pagamento
  - » Semafori sulle rampe
- | Pro e contro in entrambi gli approcci
  - » Lungo il percorso: controllo puntuale ma richiede i controllori
  - » Agli accessi: limitato ai varchi ma lascia scoperto il tracciato

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## II Flow Control

- | Il problema delle applicazioni: applicazioni differenti richiedono meccanismi differenti
- | File Transfer
  - » Per la natura burst non presenta problemi
- | CSCW
  - » Controllo di accesso preferibile al “rallentamento”
- | I parametri di QoS sono influenzati dai meccanismi di Flow Control
  - » Audio vs Video - Buffer Overflow vs Delay

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Obiettivi del Flow Control

- | E' importante ricordare che il Flow Control è solo una tecnica di gestione e controllo della rete
- | Come qualsiasi tecnica deve essere affiancata da una strategia del controllo, ossia da una serie di obiettivi prefissati
- | Esistono due aspetti che devono necessariamente essere tenuti in conto in qualsiasi strategia:
  - » Equità della politica di Flow Control
  - » Non passare da un Flow Control ad un Flow Limitation

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

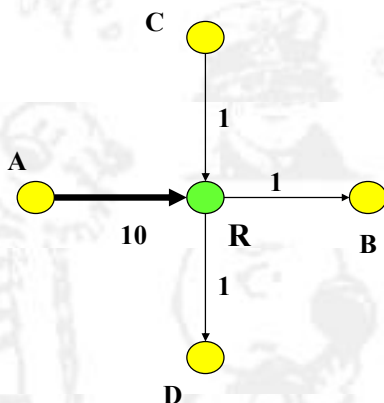
## Obiettivi del Flow Control

- | Limitare il Delay o evitare il Buffer Overflow?
- | Per le applicazioni MM o con media isocroni un delay basso è auspicabile, ed è ottenibile ottimizzando la dimensione delle code nei nodi
- | Per altre applicazioni è più importante non perdere pacchetti a causa di overflow, con possibili ritrasmissioni, e con conseguente ulteriore peggioramento delle congestioni
- | Le ritrasmissioni possono avvenire
  - » Per perdite dovute a buffer overflow
  - » Per ritardi negli ack dovuti alla lentezza

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Obiettivi del Flow Control



**A: Poisson source, rate**

**C: Poisson source, rate**

**R: Node with large, finite buffer**

**For low  $\rho$ , throughput =  $0.8 + \rho$   
For  $\rho \approx 1$ , R buffer overflows due to link RB limited capacity**

**Nodes A & C spend most time in retransmitting packets**

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Obiettivi del Flow Control

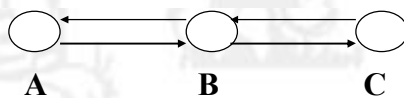
- | Il problema dell'Equità
- | Nel caso precedente, il nodo A trasmette ad una velocità 10 volte maggiore del nodo C
- | Ciò significa che ha una probabilità 10 volte maggiore di occupare uno spazio disponibile nel buffer del nodo R
- | Pertanto il throughput AB sarà circa 10 volte maggiore del throughput CD
- | Il throughput totale si assesterà intorno a 1.1

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Obiettivi del Flow Control

- | Congestioni e dead-lock
- | Qualche volta la congestione può entrare in uno stato di deadlock senza possibilità di rottura dello stallo
- | Questo può capitare quando i nodi congestionati non possono modificare il routing a causa della congestione di tutti i router vicini



**La saturazione dei buffer richiede la ritrasmissione da A a B e viceversa senza possibilità uscita dallo stallo**

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

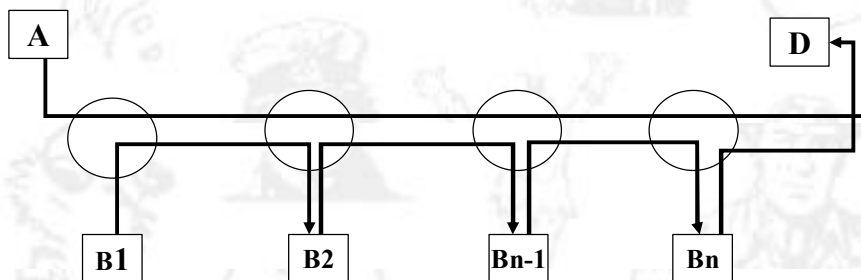
## Obiettivi del Flow Control

- | Come esercitare il controllo?
- | Nel prossimo esempio
  - » 1 Sorgente A che trasmette ad 1 unità/sec
  - » N Sorgenti Bi che trasmettono a 1 unità/sec
  - » Tutti i link hanno capacità 1 unità/sec
- | Ipotesi 1: A = 0 - Throughput totale = N (MAX!)
- | Ipotesi 2: A = Bi = 1/2 unità /sec (Accesso nodi paritetico) Throughput totale = (N + 1) / 2
- | Ipotesi 3: Bi = n / (n+1), A = 1 / (N+1) (Risorse rete divise eq.)  
Throughput totale =  $N * (N / (N+1)) + 1 / (N+1) = (N^2 + 1) / (N+1)$

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Obiettivi del Flow Control



Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Flow control problem

- | Consider file transfer
- | Sender sends a stream of packets representing fragments of a file
- | Sender should try to match rate at which receiver and network can process data
- | Can't send too slow or too fast
- | Too slow
  - » wastes time
- | Too fast
  - » can lead to buffer overflow
- | How to find the correct rate?

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Other considerations

- | Simplicity
- | Overhead
- | Scaling
- | Fairness
- | Stability
  
- | Many interesting tradeoffs
  - » overhead for stability
  - » simplicity for unfairness

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Where?

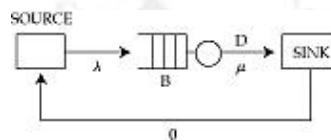
- | Usually at transport layer
- | Also, in some cases, in datalink layer

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Model

- | Source, sink, server, service rate, bottleneck, round trip time



Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Classification

- | Open loop
  - » Source describes its desired flow rate
  - » Network *admits* call
  - » Source sends at this rate
- | Closed loop
  - » Source monitors available service rate
    - Explicit or implicit
  - » Sends at this rate
  - » Due to speed of light delay, errors are bound to occur
- | Hybrid
  - » Source asks for some minimum rate
  - » But can send more, if available

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Open loop flow control

- | Two phases to flow
  - » Call setup
  - » Data transmission
- | Call setup
  - » Network prescribes parameters
  - » User chooses parameter values
  - » Network admits or denies call
- | Data transmission
  - » User sends within parameter range
  - » Network *polic*es users
  - » Scheduling policies give user QoS

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Hard problems

- | Choosing a descriptor at a source
- | Choosing a scheduling discipline at intermediate network elements
  - » This depends on the QoS scheme to be adopted (per flow, per class, per node, per network)
  - » Some time we have to stick to an existing scheme (e.g. FIFO)
- | Devising a network model
- | Admitting calls so that their performance objectives are met (*call admission control*).

## Traffic descriptors

- | Usually an *envelope*
  - » Constrains worst case behavior
- | Three uses
  - » Basis for traffic contract
  - » Input to *regulator*
  - » Input to *policer*

## Descriptor requirements

- | **Representativity**
  - » adequately describes flow, so that network does not reserve too little or too much resource
- | **Verifiability**
  - » verify that descriptor holds
- | **Preservability**
  - » Doesn't change inside the network
- | **Usability**
  - » Easy to describe and use for admission control

## Examples

- | **Representative, verifiable, but not useable**
  - » Time series of interarrival times
- | **Verifiable, preservable, and useable, but not representative**
  - » peak rate

## Some common descriptors

- | Peak rate
- | Average rate
- | Linear bounded arrival process

## Peak rate

- | Highest 'rate' at which a source can send data
- | Two ways to compute it
- | For networks with fixed-size packets
  - » min inter-packet spacing
- | For networks with variable-size packets
  - » highest rate over *all* intervals of a particular duration
- | Regulator for fixed-size packets
  - » timer set on packet transmission
  - » if timer expires, send packet, if any
- | Problem
  - » sensitive to extremes

## Average rate

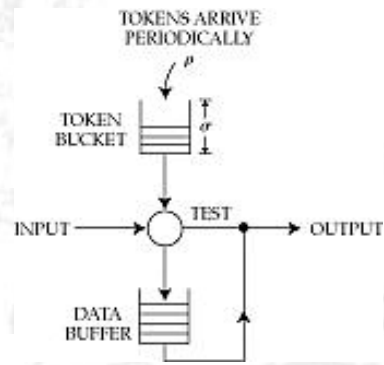
- | Rate over some time period (*window*)
- | Less susceptible to outliers
- | Parameters:  $t$  and  $a$
- | Two types: jumping window and moving window
- | Jumping window
  - » over consecutive intervals of length  $t$ , only  $a$  bits sent
  - » regulator reinitializes every interval
- | Moving window
  - » over all intervals of length  $t$ , only  $a$  bits sent
  - » regulator forgets packet sent more than  $t$  seconds ago

## Linear Bounded Arrival Process

- | Source bounds # bits sent in any time interval by a linear function of time
- | the number of bits transmitted in any active interval of length  $t$  is less than  $rt + s$
- |  $r$  is the long term rate
- |  $s$  is the burst limit
- | insensitive to outliers

## Leaky bucket

- | A regulator for an LBAP
- | Token bucket fills up at rate  $r$
- | Largest # tokens  $< s$



Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Variants

- | Token and data buckets
  - » Sum is what matters
- | Peak rate regulator

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Choosing LBAP parameters

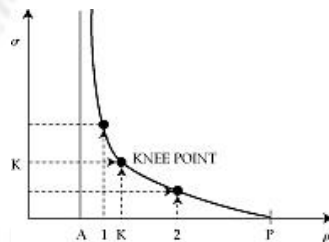
- | Tradeoff between  $r$  and  $s$
- | Minimal descriptor
  - » doesn't simultaneously have smaller  $r$  and  $s$
  - » presumably costs less
- | How to choose minimal descriptor?
- | Three way tradeoff
  - » choice of  $s$  (data bucket size)
  - » loss rate
  - » choice of  $r$

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Choosing minimal parameters

- | Keeping loss rate the same
  - » if  $s$  is more,  $r$  is less (smoothing)
  - » for each  $r$  we have least  $s$
- | Choose knee of curve



Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## LBAP

- | Popular in practice and in academia
  - » sort of representative
  - » verifiable
  - » sort of preservable
  - » sort of usable
- | Problems with multiple time scale traffic
  - » large burst messes up things

## Open loop vs. closed loop

- | Open loop
  - » describe traffic
  - » network admits/reserves resources
  - » regulation/policing
- | Closed loop
  - » can't describe traffic or network doesn't support reservation
  - » monitor available bandwidth
    - perhaps allocated using GPS-emulation
  - » adapt to it
  - » if not done properly either
    - too much loss
    - unnecessary delay

## Taxonomy

- | **First generation**
  - » ignores network state
  - » only match receiver
- | **Second generation**
  - » responsive to state
  - » three choices
    - State measurement
      - | explicit or implicit
    - Control
      - | flow control window size or rate
    - Point of control
      - | endpoint or within network

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Explicit vs. Implicit

- | **Explicit**
  - » Network tells source its current rate
  - » Better control
  - » More overhead
- | **Implicit**
  - » Endpoint figures out rate by looking at network
  - » Less overhead
- | **Ideally, want overhead of implicit with effectiveness of explicit**

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Flow control window

- | Recall error control window
- | Largest number of packet outstanding (sent but not acked)
- | If endpoint has sent all packets in window, it must wait => slows down its rate
- | Thus, window provides *both* error control and flow control
- | This is called *transmission* window
- | Coupling can be a problem
  - » Few buffers are receiver => slow rate!

## Window vs. rate

- | In adaptive rate, we directly control rate
- | Needs a timer per connection
- | Plusses for window
  - » no need for fine-grained timer
  - » self-limiting
- | Plusses for rate
  - » better control (finer grain)
  - » no coupling of flow control and error control
- | Rate control must be careful to avoid overhead and sending too much

## Hop-by-hop vs. end-to-end

- | Hop-by-hop
  - » first generation flow control at each link
    - next server = sink
  - » easy to implement
- | End-to-end
  - » sender matches all the servers on its path
- | Plusses for hop-by-hop
  - » simpler
  - » distributes overflow
  - » better control
- | Plusses for end-to-end
  - » cheaper

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## On-off

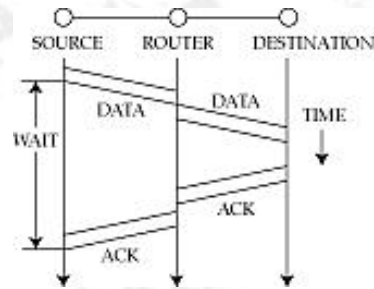
- | Receiver gives ON and OFF signals
- | If ON, send at full speed
- | If OFF, stop
- | OK when RTT is small
- | What if OFF is lost?
- | Bursty
- | Used in serial lines or LANs

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Stop and Wait

- | Send a packet
- | Wait for ack before sending next packet

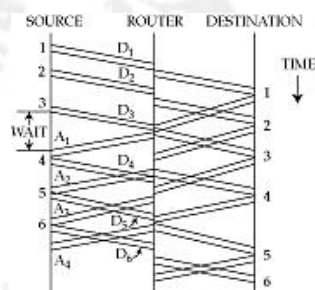


Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Static window

- | Stop and wait can send at most one pkt per RTT
- | Here, we allow multiple packets per RTT (= transmission window)



Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

What should window size be?

- | Let bottleneck service rate along path =  $b$  pkts/sec
- | Let round trip time =  $R$  sec
- | Let flow control window =  $w$  packet
- | Sending rate is  $w$  packets in  $R$  seconds =  $w/R$
- | To use bottleneck  $w/R > b \Rightarrow w > bR$
- | This is the *bandwidth delay product* or *optimal window size*

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

Static window

- | Works well if  $b$  and  $R$  are fixed
- | But, bottleneck rate changes with time!
- | Static choice of  $w$  can lead to problems
  - » too small
  - » too large
- | So, need to adapt window
- | Always try to get to the *current* optimal value

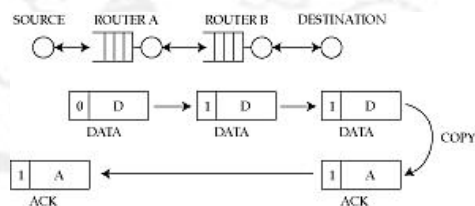
Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## DECbit flow control

### | Intuition

- » every packet has a bit in header
- » intermediate routers set bit if queue has built up => source window is too large
- » sink copies bit to ack
- » if bits set, source reduces window size
- » in steady state, oscillate around optimal size



Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## DECbit

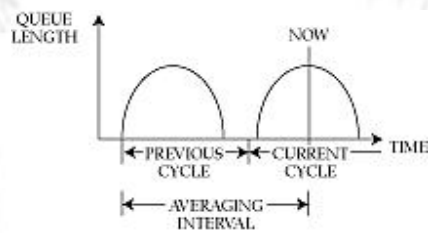
- | When do bits get set?
- | How does a source interpret them?

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## DECbit details: router actions

- | Measure *demand* and *mean queue length* of each source
- | Computed over queue regeneration cycles
- | Balance between sensitivity and stability



Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Router actions

- | If mean queue length  $> 1.0$ 
  - » set bits on sources whose demand exceeds fair share
- | If it exceeds 2.0
  - » set bits on everyone
  - » panic!

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

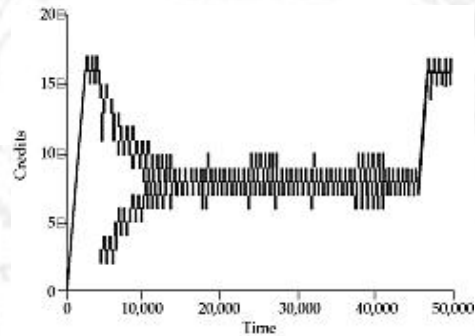
## Source actions

- | Keep track of bits
- | Can't take control actions too fast!
- | Wait for past change to take effect
- | Measure bits over past + present window size
- | If more than 50% set, then decrease window, else increase
- | Additive increase, multiplicative decrease

## Evaluation

- | Works with FIFO
  - » but requires per-connection state (demand)
- | Software
- | But
  - » assumes cooperation!
  - » conservative window increase policy

## Sample trace



Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## TCP Flow Control

- | Implicit
- | Dynamic window
- | End-to-end
  
- | Very similar to DECbit, but
  - » no support from routers
  - » increase if no loss (usually detected using timeout)
  - » window decrease on a timeout
  - » additive increase multiplicative decrease

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## NETBLT

- | First rate-based flow control scheme
- | Separates error control (window) and flow control (no *coupling*)
- | So, losses and retransmissions do not affect the flow rate
- | Application data sent as a series of buffers, each at a particular rate
- | Rate = (burst size + burst rate) so granularity of control = burst
- | Initially, no adjustment of rates
- | Later, if received rate < sending rate, multiplicatively decrease rate
- | Change rate only once per buffer => slow

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

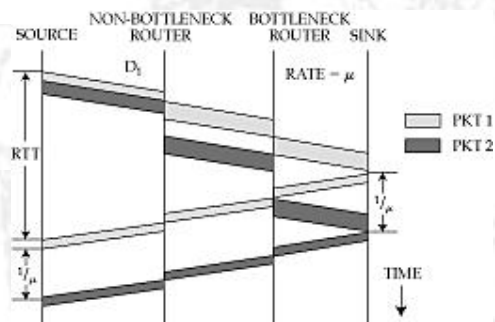
## Packet pair

- | Improves basic ideas in NETBLT
  - » better measurement of bottleneck
  - » control based on prediction
  - » finer granularity
- | Assume all bottlenecks serve packets in round robin order
- | Then, spacing between packets at receiver (= ack spacing) =  $1/(\text{rate of slowest server})$
- | If *all* data sent as paired packets, no distinction between data and probes
- | Implicitly determine service rates if servers are round-robin-like

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Packet pair



Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Packet-pair details

- | Acks give time series of service rates in the past
- | We can use this to predict the next rate
- | Exponential averager, with fuzzy rules to change the averaging factor
- | Predicted rate feeds into flow control equation

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

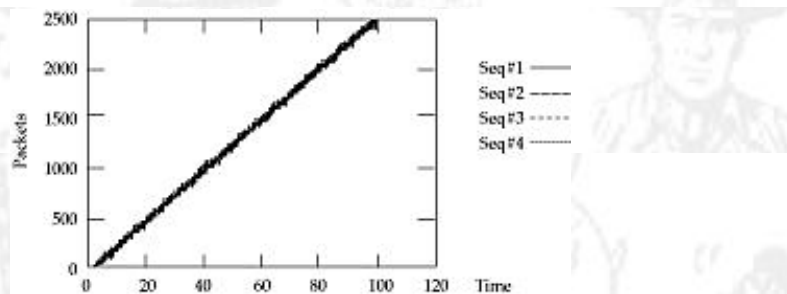
## Packet-pair flow control

- | Let  $X$  = # packets in bottleneck buffer
- |  $S$  = # outstanding packets
- |  $R$  = RTT
- |  $b$  = bottleneck rate
- | Then,  $X = S - Rb$  (assuming no losses)
- | Let  $I$  = source rate
- |  $I(k+1) = b(k+1) + (\text{setpoint} - X)/R$

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Sample trace



Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## ATM Forum EERC

- | Similar to DECbit, but send a whole cell's worth of info instead of one bit
- | Sources periodically send a Resource Management (RM) cell with a *rate request*
  - » typically once every 32 cells
- | Each server fills in RM cell with current share, if less
- | Source sends at this rate

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## ATM Forum EERC details

- | Source sends Explicit Rate (ER) in RM cell
- | Switches compute source share in an unspecified manner (allows competition)
- | Current rate = allowed cell rate = ACR
- | If  $ER > ACR$  then  $ACR = ACR + RIF * PCR$   
else  $ACR = ER$
- | If switch does not change ER, then use DECbit idea
  - » If CI bit set,  $ACR = ACR (1 - RDF)$
- | If  $ER < AR$ ,  $AR = ER$
- | Allows interoperability of a sort
- | If idle 500 ms, reset rate to Initial cell rate
- | If no RM cells return for a while,  $ACR^* = (1 - RDF)$

Corso di Reti di Calcolatori – Anno accademico 2002/2003

Dipartimento di Informatica e Sistemistica, Università di Napoli Federico II

## Comparison with DECbit

- | Sources know exact rate
- | Non-zero Initial cell-rate => conservative increase can be avoided
- | Interoperation between ER/CI switches

## Problems

- | RM cells in data path a mess
- | Updating sending rate based on RM cell can be hard
- | Interoperability comes at the cost of reduced efficiency (as bad as DECbit)
- | Computing ER is hard

## Comparison among closed-loop schemes

- | On-off, stop-and-wait, static window, DECbit, TCP, NETBLT, Packet-pair, ATM Forum EERC
- | Which is best? No simple answer
- | Some rules of thumb
  - » flow control easier with RR scheduling
    - otherwise, assume cooperation, or police rates
  - » explicit schemes are more robust
  - » hop-by-hop schemes are more responsive, but more complex
  - » try to separate error control and flow control
  - » rate based schemes are inherently unstable unless well-engineered

## Hybrid flow control

- | Source gets a minimum rate, but can use more
- | All problems of both open loop and closed loop flow control
- | Resource partitioning problem
  - » what fraction can be reserved?
  - » how?