Flow and Congestion Control

CS 218 F2003

Oct 29, 03

- Flow control goals
- Classification and overview of main techniques
- TCP Tahoe, Reno, Vegas
- TCP Westwood

- Readings: (1) Keshav's book – Chapter on Flow Control;
- (2) Stoica: Core Stateless Fair Queueing

A little bit of history

- In the **early ARPANET**, link level “alternating bit” reliable protocol => congestion protection (via backpressure), but also deadlocks!
- S/F deadlocks and Reassembly Buffer deadlocks: buffer mngt saves the day!
- In the **NPL network** (UK, early ’70s): “isarithmic” control; limit total # of packets in the net. Problems?
- Later, in X.25 networks, **link level** HDLC protocol and **network level** virtual circuit (hop-by-hop) flow control; efficient, selective backpressure to source – but, expensive
- In **ATM network**, no link level protocol needed (10Exp-9 bit error rates!); however flow control on VCs (either **hop by hop** window ctrl or **end to end rate** controlled)

A little bit of history (cont)

What about the **Internet**?

- Until ‘88 TCP with fixed window (serious problems with loss and retransmissions!!)
- In ’88, adaptive TCP window was introduced (Van Jacobsen)
- Various “improved” versions: Tahoe, Reno, Vegas, New Reno, Snoop, Westwood, Peach etc (1996 to 2002)
- Recently, the strict “end to end” TCP paradigm has been relaxed: first, **Active Queue Management**; then, explicit network feedback (**Explicit Network Notification** – of congestion; **XCP**, eXplicit Control Protocol)
- **Hybrid** end to end and network feedback model

Flow Control - the concept

- **Flow Control**: “ set of techniques which match the **source offered rate** to the available **service rate** in the network and at the receiver..”
- **Congestion Control**: “..techniques preventing network buffers overflow”
- **Design Goals** (best effort flow/congestion control):

Efficient (low O/H; good resource utilization)

Fair (ie, max-min fair)

Stable (converges to equilibrium point; no intermittent “capture”)

Scaleable (eg, limit on per flow processing O/H)

Flow Control - Classification

Open loop flow control - guaranteed service:

- user declares traffic descriptor/ Qos Parameters
- call admission control (CAC); QoS negotiation
- network reserves resources (bdw, buffers)
- user “shapes”; network “policies” (eg, Leaky Bucket)
- another example: real time stream layer shadding

Open loop flow control - best effort :

- user does not declare traffic descriptors/QoS
- network drops packets to enforce Fair Share among best effort sources (eg, Core-stateless Fair Sharing)

Flow Control - Classification (cont)

Closed loop flow control:

- *best effort*: eg, TCP; or ATM PRCA (Prop Rate Contr Alg)
- *real time adaptive* QoS (eg, adaptive source encoding)
- **concept**: network feedback (explicit or implicit) forces the user to adjust the offered rate
- control strategy at source may vary: adaptive window; adaptive rate; adaptive code; layer shadding, etc

Hybrid open and closed loop:

- **min QoS** (eg, bandwidth) guarantee + **best effort** resource allocation above minimum (eg, "ABR +" in ATM)

Flow Control vs Congestion Control

Traditional interpretation (as seen before):

- **flow control** = end to end flow control
- **congestion control** = control at intermediate nodes

However, the distinction is *fuzzy*:

- example: Hop by Hop flow control on VCs (as in X.25) operates at intermediate nodes but indirectly has end to end impact via **backpressure**
- **alternate definition**: congestion control operates on internal flows without discriminating between source and sink (under this definition, VC-FC is "flow control")

Closed Loop Control ("Hop by Hop")

Non selective hop by hop "**congestion control**":

+ efficient; incorporated in popular Data Link protocols (eg, HDLC, SDLC etc); predominant in the old ARPANET

- unfair; may lead to **deadlocks**

Selective (per flow) hop by hop "**flow control**":

+ very effective; induces backpressure; fair

- "per-flow" does not scale well; excessive O/H

Internet does **not use** Link Level Congestion/Flow control: PPP and E-nets have no flow control. ATM VCs tunnel IP traffic over the ATM. But, they use UBR or CBR service (no flow control)

Open Loop control

- **traffic descriptors**: peak rate, avg rate, burst length
- **traffic contract**; QoS negotiation; CAC
- **regulator at user side: “shaper”**, smoother (delays abusive packets)
- **regulator at network side: “policer”** (drops/marks packets violating the traffic contract)
- examples of traffic regulators:

peak rate: enforces inter packet spacing (fixed size pkts)

average rate: (a) **jumping window** (rate estimation over consecutive windows); (b) **moving window** (estimation over a sliding window)

Open Loop Control- traffic descriptors

- **Linear Bounded Arrival Process** (LBAP):

  # of bits NB transmitted in any interval t:

  $NB = rt + s$

  r = long term average rate

  s = longest burst sent by source
- **Leaky Bucket**: regulator for 2-parameter LBAP
- **Design Issue**: many possible (r,s) pairs for a source; how to select the **"minimal" LBAP descriptors ?** Knee.. **Problems**: dynamic changes in traffic/service parameters; long range dependence. Solution: **renegotiation**

Closed Loop Schemes - Classification

- Used for **best effort** sources (no reservations). The **classification** can be based on the following features:
- (a) **Implicit vs Explicit state measurement**: user must infer available resources, or network specifically tells
- (b) **Dynamic Window vs rate** adaptation: eg, TCP window; ATM source rate control
- (c) **Hop-by-hop vs end-to-end**: HbH more responsive to network state feedback than EtE (may use both, like in ECN for TCP)

Rate Based schemes

Explicit state:

- ATM Forum EERC; Smith Predictor PRCA
- Mishra/Kanakia

Implicit state

- Packet-Pair

ATM Forum EERC

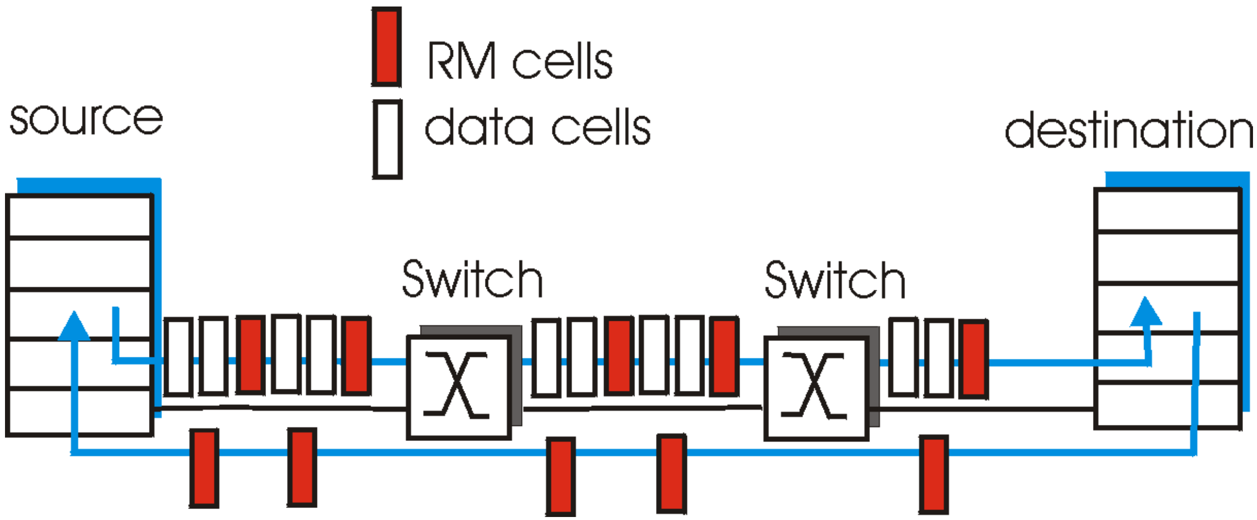
- EERC: End to End Rate Control
- Control of ABR traffic (Available Bit Rate)
- Source transmits one RM (Resource Mngt) cell every NRM (Non RM) cells (typically, NRM = 32)
- RM carries Explicit Rate (ER): the proposed rate
- Intermediate switches dynamically compute **Fair Share** and reduce ER value accordingly (FS computation not specified by ATM Forum)
- RM returns to source with reduced ER

ATM ABR congestion control

RM (resource management) cells:

- bits in RM cell set by switches ("*network-assisted*")
  - NI bit: no increase in rate (mild congestion)
  - CI bit: congestion indication
- RM cells returned to sender by receiver, with bits intact

ATM ABR congestion control



- two-byte ER (explicit rate) field in RM cell
  - congested switch may lower ER value in cell
  - sender' send rate thus minimum supportable rate on path
- EFCI bit in data cells: set to 1 in congested switch
  - if some data cell preceding the RM cell has EFCI set, then the receiver sets the CI bit in returned RM cell

EERC: Source Behavior

At VC set up, negotiation of:

- **Min Cell Rate** (guaranteed by the network);
- **Peak CR** (not to be exceeded by source);
- **Initial CR** (to get started)

ACR (Allowed CR), is dynamically adjusted at source:

If ER > ACR

ACR = ACR + RIF * PCR (additive increase)

Else, If ER < ACR

ACR = ER

EERC - extensions

- To enable interoperation with switches which cannot compute Fair Share, the RM cell carries also CI (Congestion Indication) bit in addition to ER
- Source reacts differently if CI = 1 is received

  ACR = ACR (1-RDF) multiplicative decrease

  If ACR > ER, then ACR = ER
- For robustness: if source silent for 500ms, ACR is reset to ICR; if no RMs returned before T/Out, multipl decrease
- Problem: computation of Fair Share is complex (need to measure traffic on each flow)

Mishra-Kanakia Hop by Hop Rate Control

- Rate computed at each hop based on downstream neighbor feedback
- Each node periodically sends to upstream neighbor the sampled **service rate** and **buffer occupancy** for each flow (note: all flows have same buffer **target threshold B)**
- Upstream node computes own service rate as follows:
- predicts downstream node service rate (exp average) and buffer occupancy for each flow
- computes own rate so as to approach the buffer threshold B

Mishra-Kanakia (cont)

- Scheme achieves max-min fairness (because of common buffer threshold B)
- Reacts more promptly than end to end rate control (can achieve equilibrium in 2 round trip times)
- No round robin scheduling required
- However, per flow rate estimation quite complex!

Packet-Pair (Keshav)

- **Rate** based; **implicit** state
- round robin, per-flow scheduling at routers
- packets are transmitted by pairs: the time gap between ACKs allows to **estimate bottleneck rate**, say u(k), at time k at the source
- next, compute bottleneck **buffer occupancy** X:

  $X = S - u(k)\,RTT$

  where S = # of outstanding, un-ACKed pkts

Packet-Pair (cont)

- Select new tx rate l (k+1) such that the buffer occupancy can achieve a **common target B**:

  l (k+1) = u (k) + (B - X)/RTT

- in essence, the goal is to keep the bottleneck queues at the same level using the rate measurement as feedback
- scheme is **max-min fair and stable**;
- it cleverly decouples error control (window) from flow control (rate)
- implementation drawback: **per-flow scheduling**!

Dynamic Window Control

- **Credit based hop by hop scheme**: used in X25 VCs, and proposed (unsuccessfully) for ATM ABR control
- **DECbit**: **end to end** scheme (like TCP). It uses **explicit** queue measurements at routers (with DECbit feedback) to adjust the send window
- **TCP: end to end**. It uses **implicit** feedback (packet loss) to infer buffer congestion