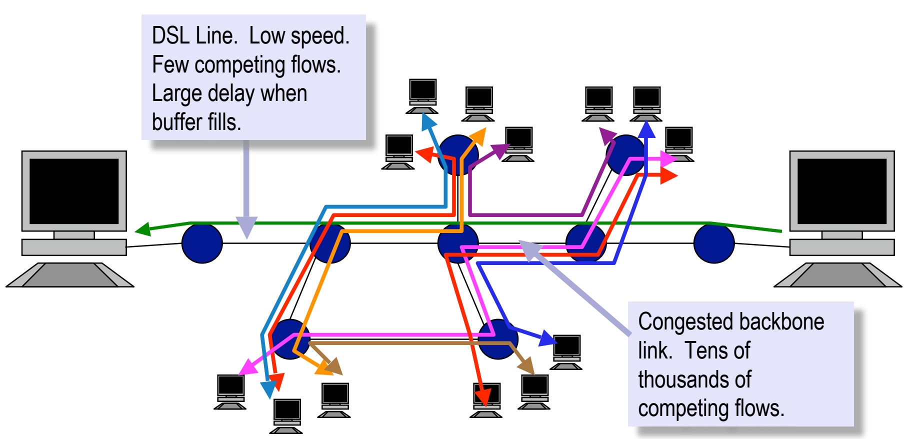
16: Congestion Control

Mark Handley

What happens in the net?

- Your traffic gets mixed with lots of other traffic.



Loss and Jitter.

- At each router, there is a queue.
  - Even if the link is not very congested, sometimes incoming packets arrive simultaneously for the same outgoing link.
- When incoming traffic exceeds outgoing link speed, a queue builds.
  - Packets are delayed - jitter.
  - When the queue fills, packets are dropped.
- Occasionally routes change.
  - Sometimes lose packets.
  - Delay changes.
- Wireless links may do link-layer retransmission to reduce loss.
  - Adds to jitter.

Main issue though is competing traffic in the same queue.

Outline

- Purpose of congestion control
- TCP's congestion control algorithm (AIMD)
- Suitability of AIMD for multimedia
- TCP-friendly congestion control for multimedia
  - TCP modelling
- Applications and Congestion Control
- Datagram Congestion Control Protocol (DCCP)

Congestion Control

End-to-end congestion control serves several purposes:

- Divides bandwidth between network flows in a "reasonably fair" manner without requiring per-flow scheduling by routers.
- Prevents congestion collapse of the network by matching demand to supply to ensure overall goodput remains reasonably high.

Congestion Collapse

Congestion collapse occurs when the network is increasingly busy, but little useful work is getting done.

Problem: *Classical congestion collapse*:

Paths clogged with unnecessarily-retransmitted packets [Nagle 84].

Fix:

Modern TCP retransmit timer and congestion control algorithms [Jacobson 88].

Fragmentation-based congestion collapse

Problem:

Paths clogged with fragments of packets invalidated because another fragment (or cell) has been discarded along the path. [Kent and Mogul, 1987]

Fix:

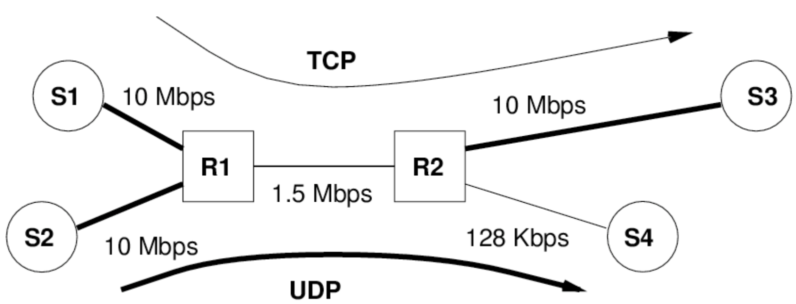
MTU discovery [Mogul and Deering, 1990]

Early Packet Discard in ATM networks [Romanow and Floyd, 1995].

Congestion collapse from undelivered packets

Problem: Paths clogged with packets that are discarded before they reach the receiver [Floyd and Fall, 1999].

Fix: Either end-to-end congestion control, or a ``virtual-circuit'' style of guarantee that packets that enter the network will be delivered to the receiver.



Congestion Control

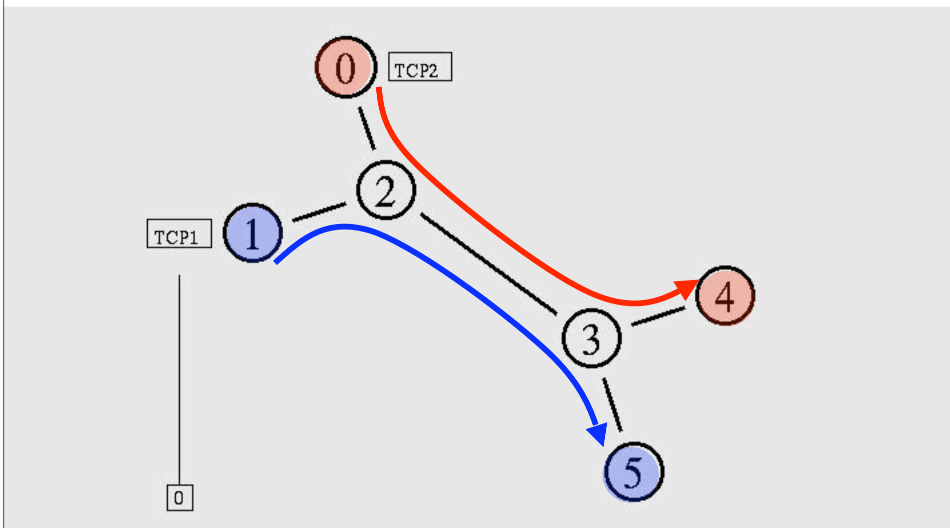
Since 1988, the Internet has remained functional despite exponential growth, routers that are sometimes buggy or misconfigured, rapidly changing applications and usage patterns, and flash crowds.

This is largely because most applications use TCP, and TCP implements *end-to-end congestion control.*

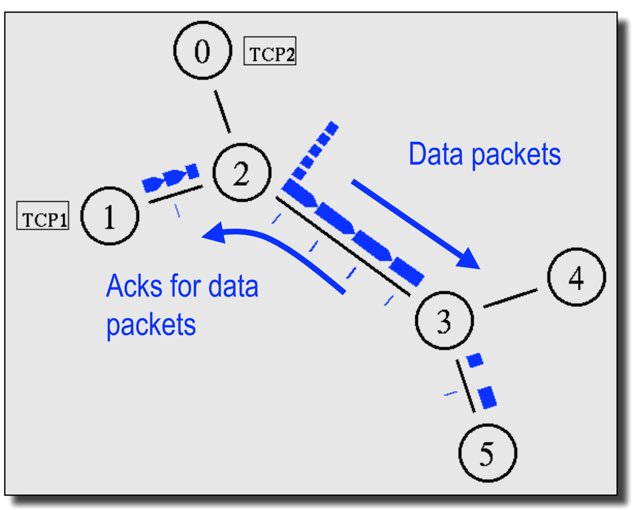
Outline

- Purpose of congestion control
- TCP's congestion control algorithm (AIMD)
- Suitability of AIMD for multimedia
- TCP-friendly congestion control for multimedia
  - TCP modelling
- Applications and Congestion Control
- Datagram Congestion Control Protocol (DCCP)

TCP tries to answer the question “how fast should I send?”

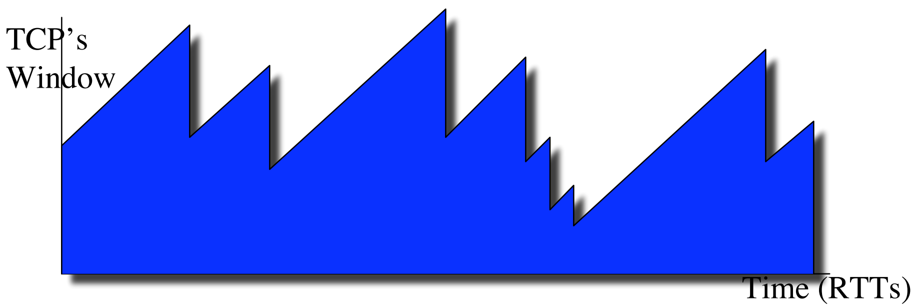


TCP’s window is all the packets TCP has sent for which it has not yet seen the acknowledgment.

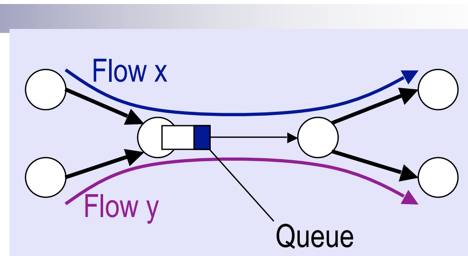


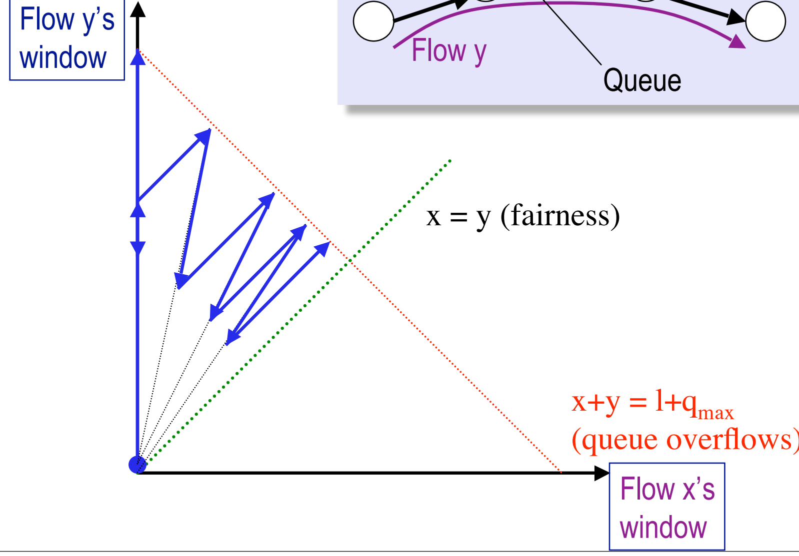
TCP's congestion control adapts the window to fit the capacity available in the network.

- Each round-trip time, increase window by one packet.
- If a packet is lost, halve the window.

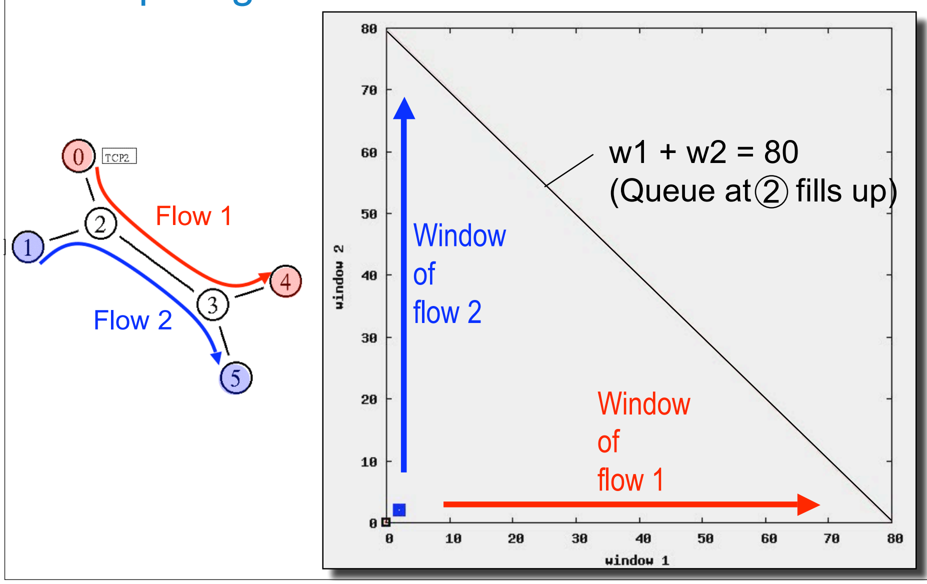


TCP Fairness

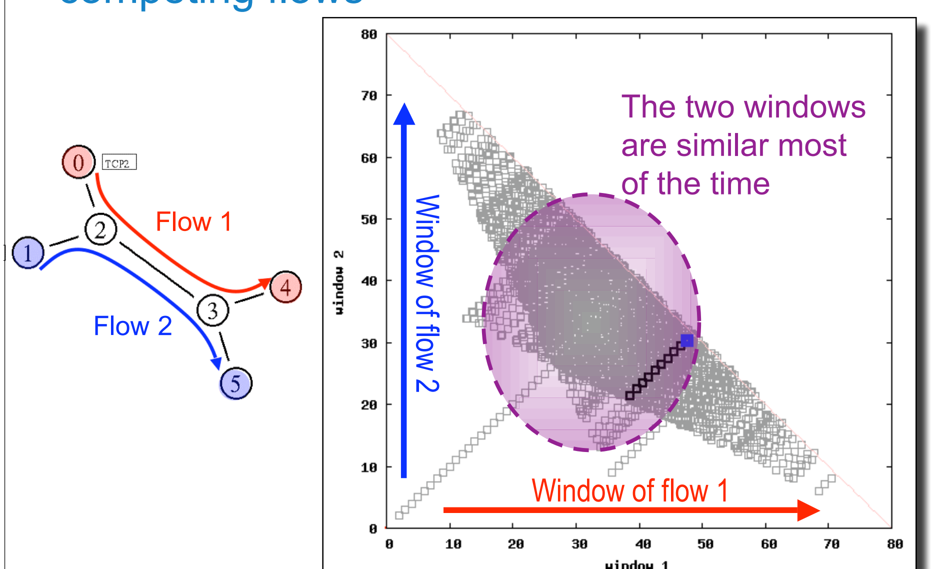




Over time, TCP equalizes the windows of competing flows



Over time, TCP equalizes the windows of competing flows



TCP (Details)

- TCP congestion control uses AIMD:
  - Increase the congestion window by one packet every round-trip time (RTT) that no packet is lost.
  - Decrease the congestion window by half every RTT that a packet loss occurs.
- In heavy congestion, when a retransmitted packet is itself dropped or when there aren't enough packets to run an ACK-clock, use a retransmit timer, which is exponential backed off if repeated losses occur.
- Slow-start: start by doubling the congestion window every roundtrip time.

The Future

TCP continues to serve us well as the basis of most transport protocols, but some important applications are not well suited to TCP:

- Telephony and Video-telephony.
- Streaming Media.
- Multicast Applications.

TCP is a reliable protocol. To achieve reliability while performing congestion control means trading delay for reliability.

- Telephony and streaming media have limited delay budgets - they don't want total reliability.

Non-TCP Congestion Control.

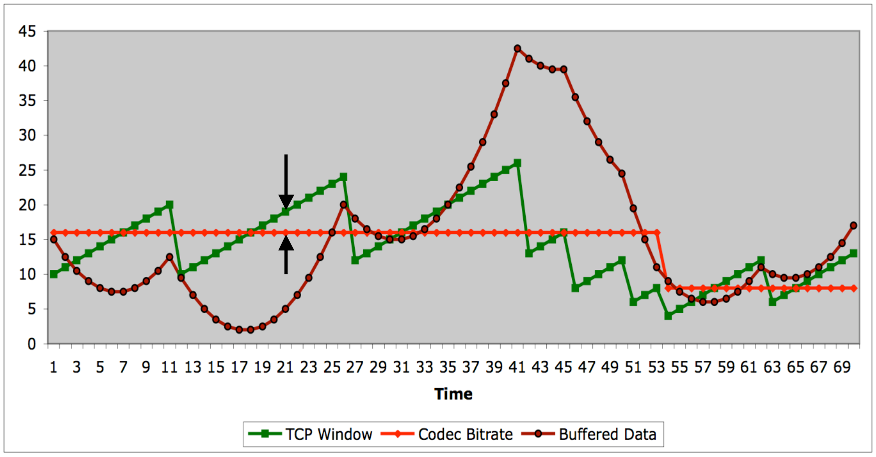
We can separate TCP's congestion control (AIMD) from TCP's reliability mechanism.

- Eg: RAP (Rate Adaptation Protocol) Rejaie et al, Infocom 1999.

However, AIMD congestion control gives a flow throughput that *changes very rapidly*, which is not well suited to streaming applications that want to delivery consistent quality to the end-user.

- Streaming playback from servers can work around this using *receiver buffering* (Eg: Rejaie et al, Sigcomm 1999), but it would be better to have a congestion control scheme that was less variable in the first place.

AIMD and Streaming Media



Send rate - consumption rate = rate of change of buffered data

Outline

- Purpose of congestion control
- TCP's congestion control algorithm (AIMD)
- Suitability of AIMD for multimedia
- TCP-friendly congestion control for multimedia
  - TCP modelling
- Applications and Congestion Control
- Multicast congestion control.
- Datagram Congestion Control Protocol (DCCP)

TCP-Friendly

- Any alternative congestion control scheme needs to *coexist with TCP in FIFO queues* in the best-effort Internet, or *be protected from TCP* in some manner.
- To co-exist with TCP, it must impose the same long-term load on the network:
  - No greater long-term throughput as a function of packet loss and delay so TCP doesn't suffer
  - Not significantly less long-term throughput or it's not too useful

Solution Space: Unicast Streaming Media

1. AIMD with different constants
   - Eg: increase window by $^3/_7$ packets each RTT, decrease multiplicatively by $^3/_4$ when a loss occurs.

2. Equation-based congestion control.
   - Try to design a protocol to achieve the throughput as TCP does on medium timescales.

TFRC: General Idea

Use a model of TCP's throughout as a function of the loss rate and RTT directly in a congestion control algorithm.

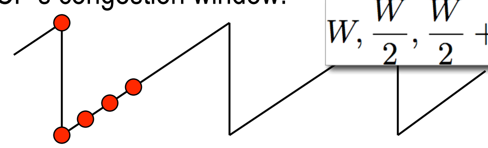
- If transmission rate is higher than that given by the model, reduce the transmission rate to the model's rate.
- Otherwise increase the transmission rate.

TCP Modelling: The "Steady State" Model

The model: Packet size B bytes, round-trip time R secs, no queue.

- A packet is dropped each time the window reaches W packets.
- TCP's congestion window:

$$W, \frac{W}{2}, \frac{W}{2}+1, \frac{W}{2}+2, \ldots, W, \frac{W}{2}, \ldots$$



- The maximum sending rate in packets per roundtrip time: W
- The maximum sending rate in bytes/sec: $W\,B\,/\,R$
- The average sending rate T: $T = (3/4)W\,B\,/\,R$
- The packet drop rate p: $p = \frac{1}{\frac{3}{8}W^2}$
- **The result:** $T = \frac{\sqrt{6}B}{2R\sqrt{p}} = \frac{\sqrt{3/2}B}{R\sqrt{p}}$

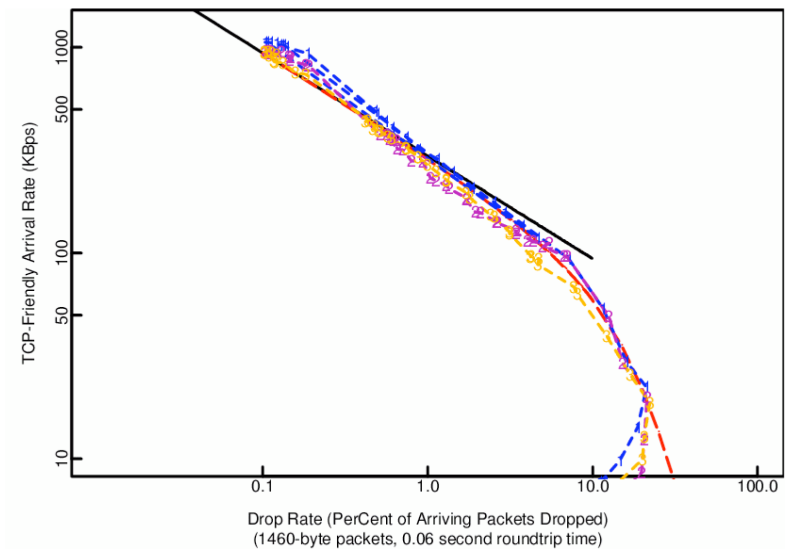
An Improved "Steady State" Model

A pretty good improved model of TCP Reno, including timeouts, from Padhye et al, Sigcomm 1998:

$$T = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}(3\sqrt{\frac{3p}{8}})p(1+32p^2)}$$

T : sending rate in bytes/second]
R : round trip time
p : fraction of packets lost
t_{RTO} : TCP retransmission timeout

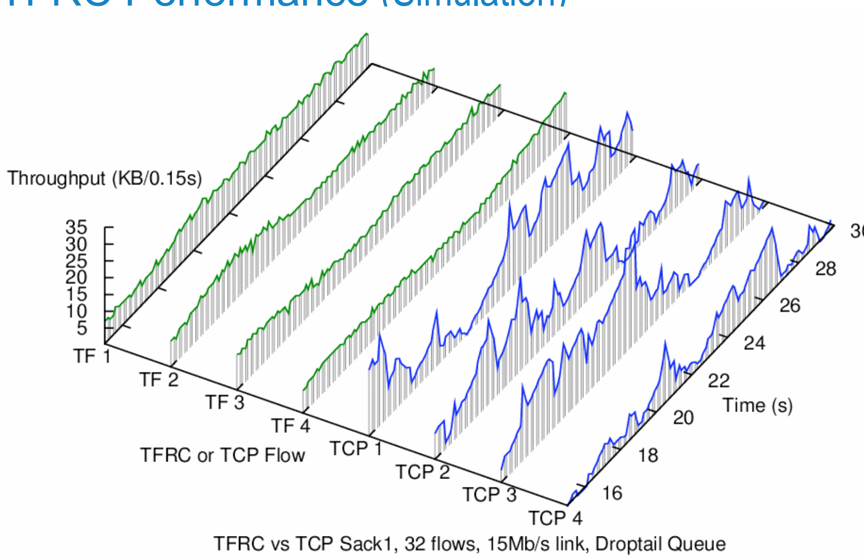
Verifying the Models



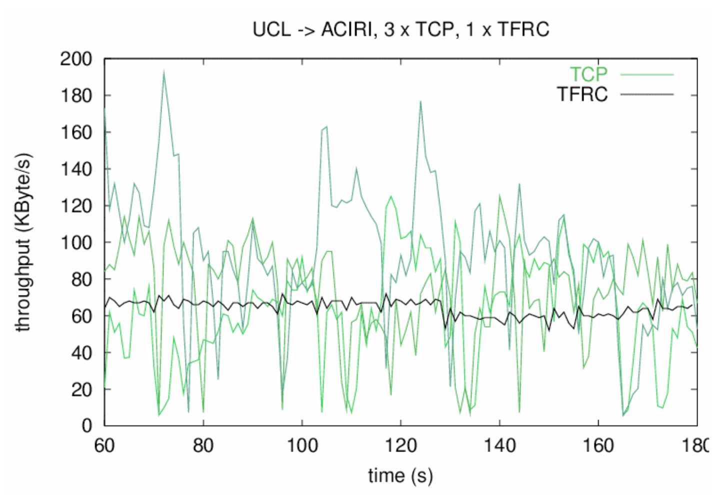
TFRC Details

- The devil's in the details.
  - How to measure the loss rate?
  - How to use RTT and prevent oscillatory behavior?
- Not as simple as we first thought.
- For the details, see:
  - Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widmer, *Equation-Based Congestion Control for Unicast Applications,* Proc ACM SIGCOMM 2000.

TFRC Performance (Simulation)



TFRC Performance (Experimental)



Outline

- Purpose of congestion control
- TCP's congestion control algorithm (AIMD)
- Suitability of AIMD for multimedia
- TCP-friendly congestion control for multimedia
  - TCP modelling
- Applications and Congestion Control
- Datagram Congestion Control Protocol (DCCP)

Applications and Congestion Control

- *"What's in it for me?"*
  - Why would an multimedia application writer choose to add congestion control?
  - Disadvantages:
    - Extra Complexity.
    - Get to go slower.
    - Variable quality may annoy users.
- "*I can just add all this redundancy and FEC you told me about to protect my flows from packet loss.*"
- *"If I don't adapt my rate, all those adaptive TCP flows will just be nice and get out of my way!"*

Internet Architecture Board is worried too.

"IAB Concerns Regarding Congestion Control for Voice Traffic in the Internet

Authors:	*S. Floyd, J. Kempf*
Filename:	*RFC 3714*
Date:	*2004-3-26*

This document discusses IAB concerns about effective end-to-end congestion control for best-effort voice traffic in the Internet. These concerns have to do with fairness, user quality, and with the dangers of congestion collapse on underprovisioned links in the Internet."

Applications and Congestion Control

Advantages of congestion control for multimedia:

- Lower packet loss (at least on lightly shared links)
- Gives ECN a chance to work.
  - Maybe no packet loss.
- Robust to network problems.
  - Won't just fall in a heap when things go wrong.
  - Won't stop your payroll from being processed!
- Get to go faster when things work well
  - CD-quality phone calls?
- Don't need to ask the user if their Internet connections is a modem/T1/DSL/Ethernet.
  - Mobile devices often don't know in advance.

Application-level Adaptation

- Streaming media options:
  - Switch between different quality stored encodings.
  - Use a layered codec, and add/drop layers as required.
  - Store high quality, and transcode as required.
- Video telephony options (easy):
  - Change quantization, frame-rate, image size.
- Audio telephony options (hard):
  - Codec switching (not ideal - audible artifacts).
  - A few codecs support adjustable bitrates (but not many)

Audio Adaptation for Telephony

- Usually very hard to track the exact rate given by the congestion control algorithm.
  - *This area is still research.*
- My current recommendation:
  - Use the congestion controlled rate as the reference rate.
  - Aim to stay within a factor of two of the reference rate.
  - Track correlation between loss rate and transmit rate in case you're the only flow on the bottleneck link!

Future Research

- Audio congestion control.
  - TFRC, TCP, and RAP all assume you change the packet rate (leaving the packet size constant).
  - What if you need to change the packet size (compression ratio) and leave the packet rate constant?

Outline

- Purpose of congestion control
- TCP's congestion control algorithm (AIMD)
- Suitability of AIMD for multimedia
- TCP-friendly congestion control for multimedia
  - TCP modelling
- Applications and Congestion Control
- Datagram Congestion Control Protocol (DCCP)

Datagram Congestion Control Protocol (DCCP)

- Implementing congestion control correctly is hard.
- It's not usually the area of expertise of the application writer, and certainly doesn't get their product to market faster.
- TCP is a non-starter.
- UDP has problems getting though firewalls and NATs because it's connectionless.

"How about providing a protocol to help out the application writers, and give them some incentive to do the right thing?"

- Result: DCCP.

DCCP Features

"DCCP provides the following features:

- *An unreliable flow of datagrams, with acknowledgements.*
- *A reliable handshake for connection setup and teardown.*
- *Reliable negotiation of options, including negotiation of a suitable congestion control mechanism.*
- *Mechanisms allowing a server to avoid holding any state for unacknowledged connection attempts or already-finished connections.*
- *An optional mechanism that allows the sender to know, with high reliability, which packets reached the receiver.*
- *Congestion control incorporating Explicit Congestion Notification (ECN) and the ECN Nonce.*
- *Path MTU discovery, as per [RFC 1191]."*

Typical DCCP connection progression

```
Client State:                                    Server State:
-------------                                    -------------
CLOSED                                             LISTEN
REQUEST              DCCP-Request
                     DCCP-Response                 RESPOND
OPEN                 DCCP-Ack
                     DCCP-Data                     OPEN
                     DCCP-Ack
                      ....
                     DCCP-Data
                     DCCP-Ack
                       ...
                     DCCP-CloseReq                 CLOSEREQ
CLOSING              DCCP-Close
                     DCCP-Reset                    CLOSED
TIME-WAIT
CLOSED
    The client and server's typical progress through states.
```

DCCP Generic Header

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |           Dest Port           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data Offset  | CCVal | CsCov |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Res | Type  |X|                 Sequence Number               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Additional ACK header

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Reserved    |            Acknowledgement Number             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

DCCP Congestion Control

DCCP supports negotiation of the congestion control mechanism. Two CCIDs currently specified:

CCID 2: TCP-like congestion control.

- AIMD without TCP's reliability
- For applications that can tolerate AIMD's sawtooth behaviour and rapid changes in bandwidth.
- Advantages: rapid reaction lowers loss rate, quickly takes advantage of available capacity.

CCID 3: TFRC congestion control.

- For applications where smoothness and predictability is most important.

DCCP status

- RFC 4340.
- Currently a few implementations, but none ready for primetime.
  - Linux probably the best; partly written by Andrea Bittau (UCL)
- Operating system APIs still a work-in-progress.
- Expect a few years before it is commonplace enough for application writers, firewalls and NATs to assume it's existence.

References

- J. Nagle, **Congestion Control in IP/TCP Internetworks**, RFC896, 1984.
- C. A. Kent, J. C. Mogul, **Fragmentation Considered Harmful**, WRL Technical Report 87/3, DEC Western Research Laboratory, Palo Alto, CA, 1987.
- V. Jacobson, **Congestion Avoidance and Control**, Proc. ACM SIGCOMM 1988
- J.C. Mogul, S.E. Deering **Path MTU discovery,** RFC1191, 1990.
- Padhye, J., Firoiu, V., Towsley, D. and J. Kurose, **Modeling TCP Throughput: A Simple Model and its Empirical Validation**, Proc. ACM SIGCOMM 1998.
- Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widmer, **Equation-Based Congestion Control for Unicast Applications**, Proc ACM SIGCOMM 2000.
- M. Handley, S. Floyd, J. Padhye, J. Widmer, **TCP Friendly Rate Control (TFRC): Protocol Specification,** RFC3448, January 2003.
- E. Kohler, M.Handley, S. Floyd, **Datagram Congestion Control Protocol (DCCP),** RFC 4340, March 2006.