# Congestion Control for Real-time Media: *History and Problems*

**Mark Handley**, UCL

# What is Congestion Control all About?

- Bulk Transfer
  - Goal is to transfer *n* bytes in zero time.

    (subject to a few minor limitations of the hardware)
- Implication
  - If the network isn't congested when doing bulk transfer, something is broken.
  - Congestion is normal.

# Primary Goals of Congestion Control

(from a network point of view)

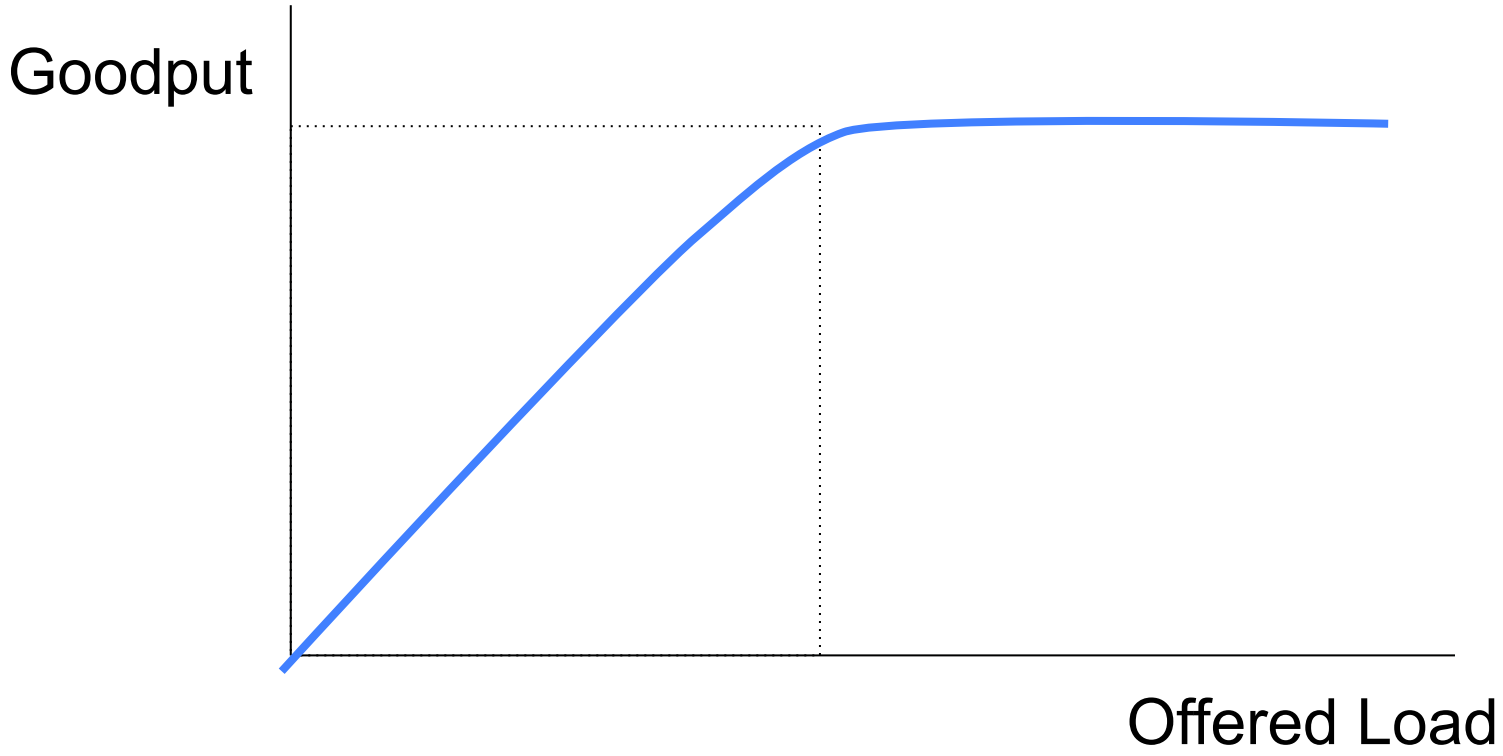
1. Avoid congestion collapse
   - Network must work.
2. Some sort of fairness
   - All users must get some service.

# Primary Goals of Congestion Control

(from a network point of view)

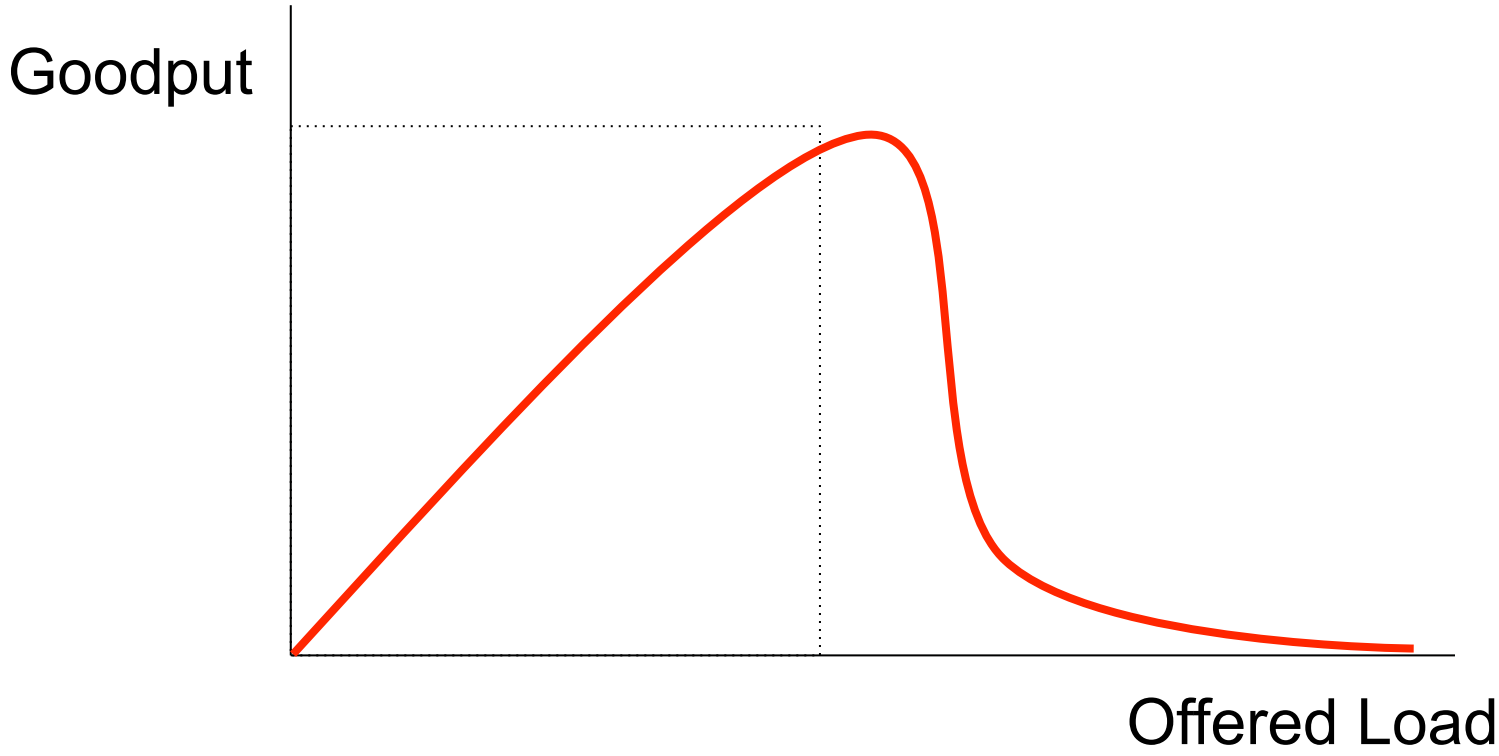
1. Avoid **congestion collapse**
   - Network must work.
2. Some sort of **fairness**
   - All users must get some service.

# Congestion Behaviour



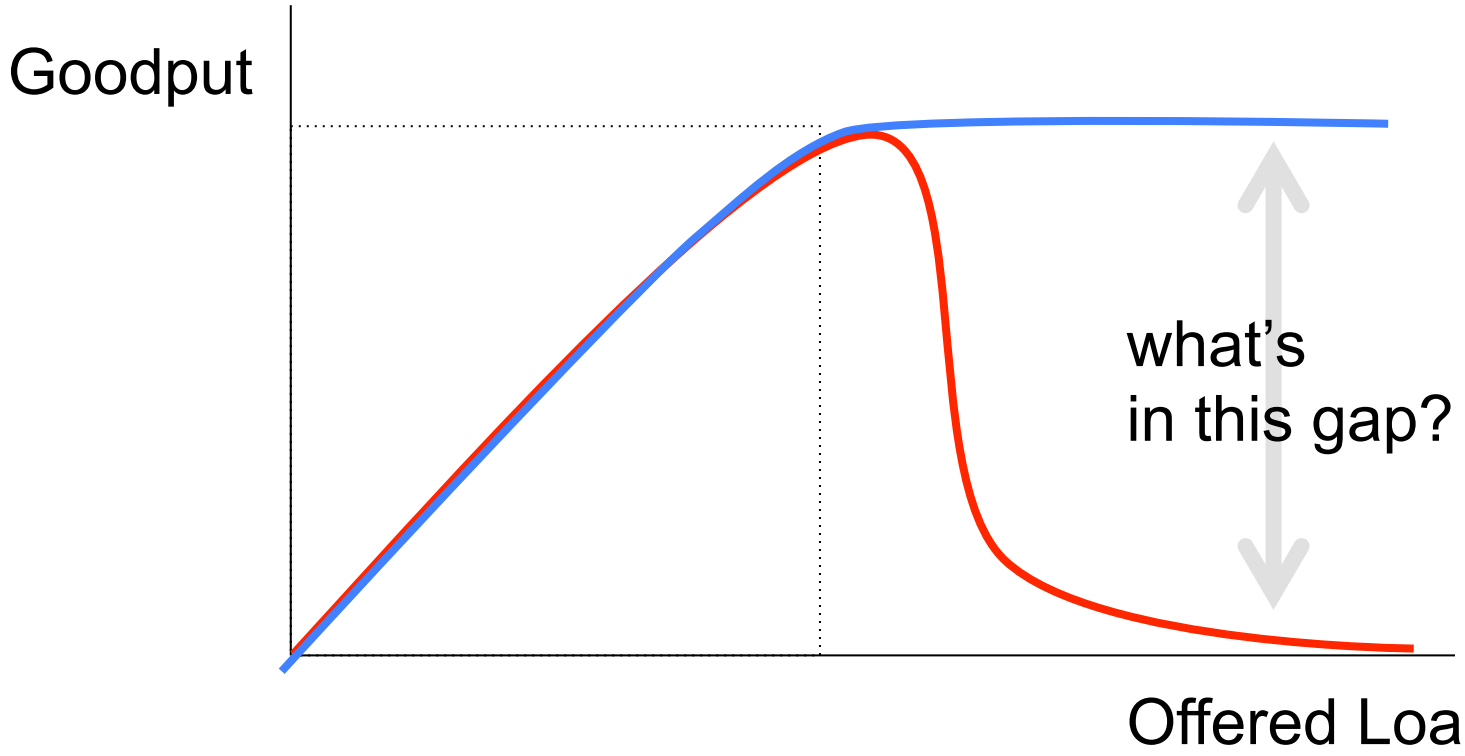

- Desired behaviour: goodput saturates at network capacity

# Congestion Collapse



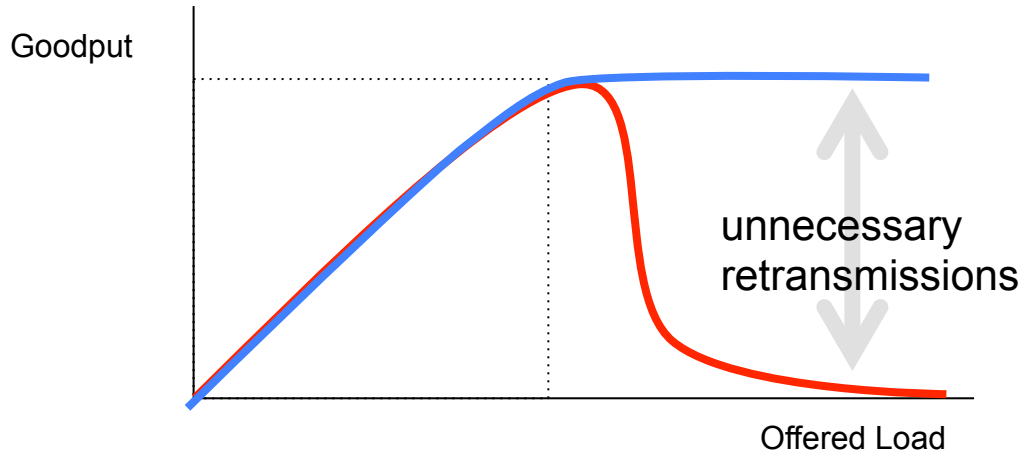

- Goodput decreases as network becomes overloaded

# Congestion Collapse




- Goodput decreases as network becomes overloaded

# Congestion Collapse



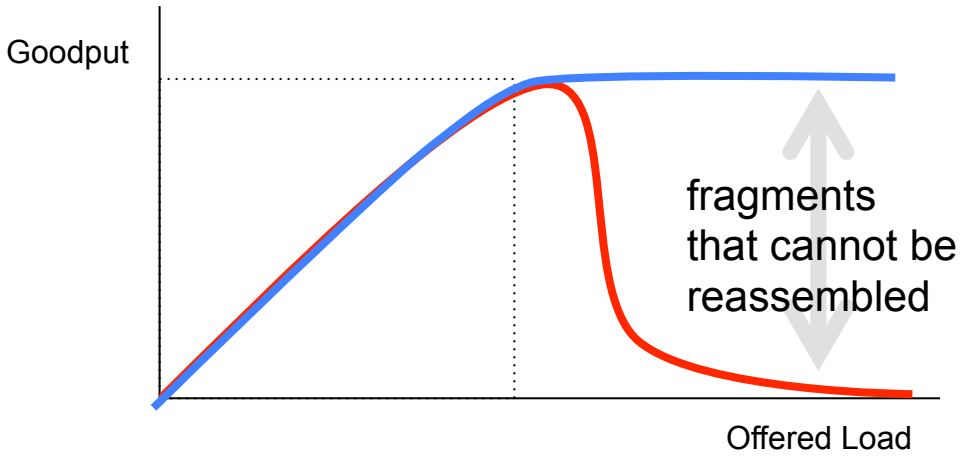

Problem: *Classical congestion collapse*:

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:

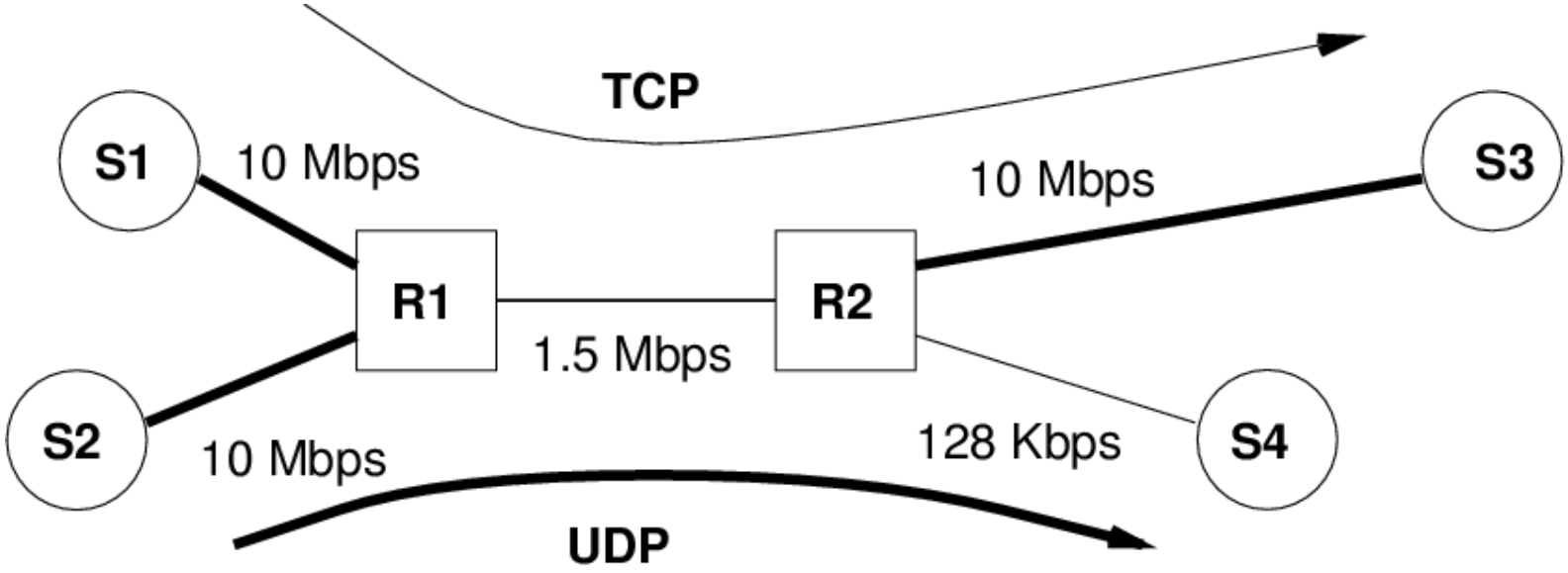
MTU discovery [Mogul and Deering, 1990]

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

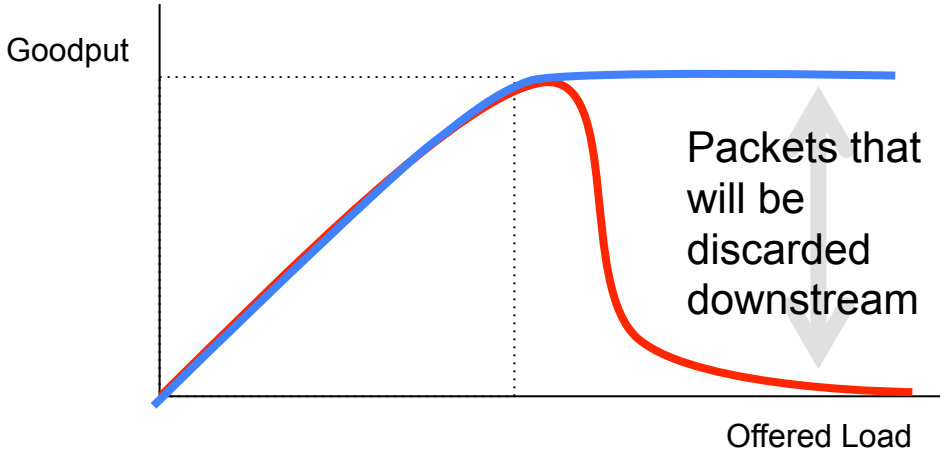
# Congestion collapse from undelivered packets

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

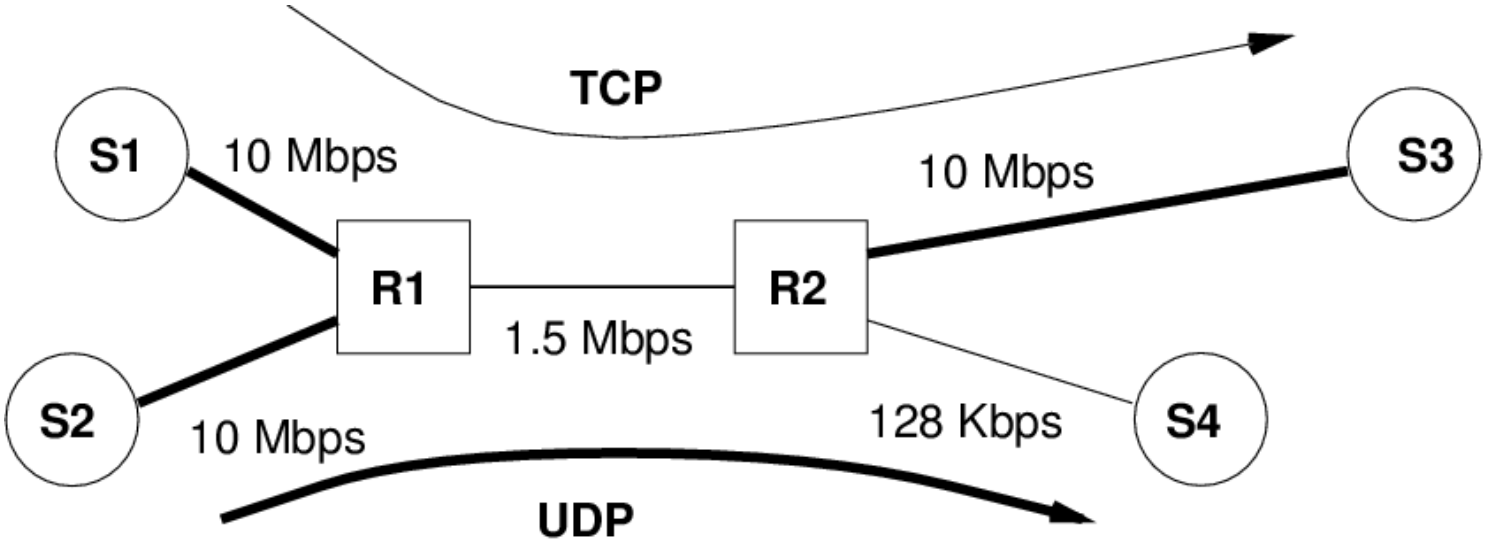
**Fix:** Either end-to-end congestion control, or a ``virtual-circuit'' style of guarantee that packets that enter the network will be delivered to the receiver.



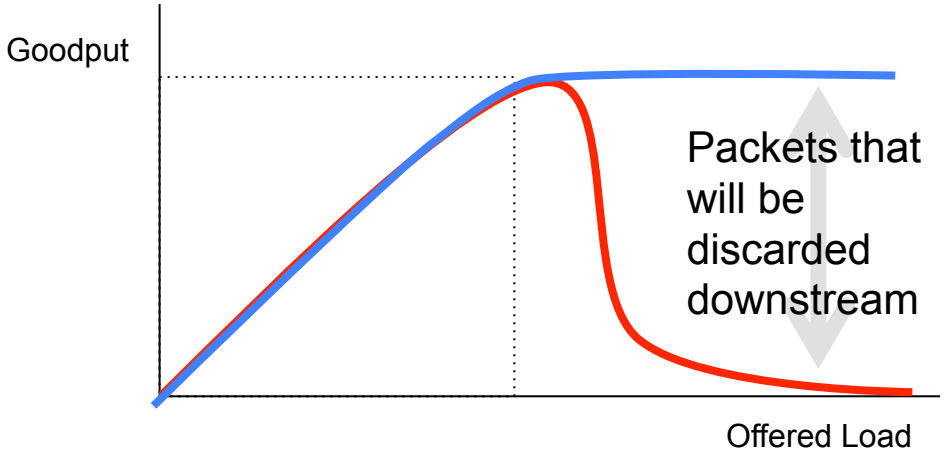
# Congestion collapse from undelivered packets




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



# 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
- ``virtual-circuit'' style of guarantee that packets that enter the network will be delivered to the receiver.

UDP

# 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.

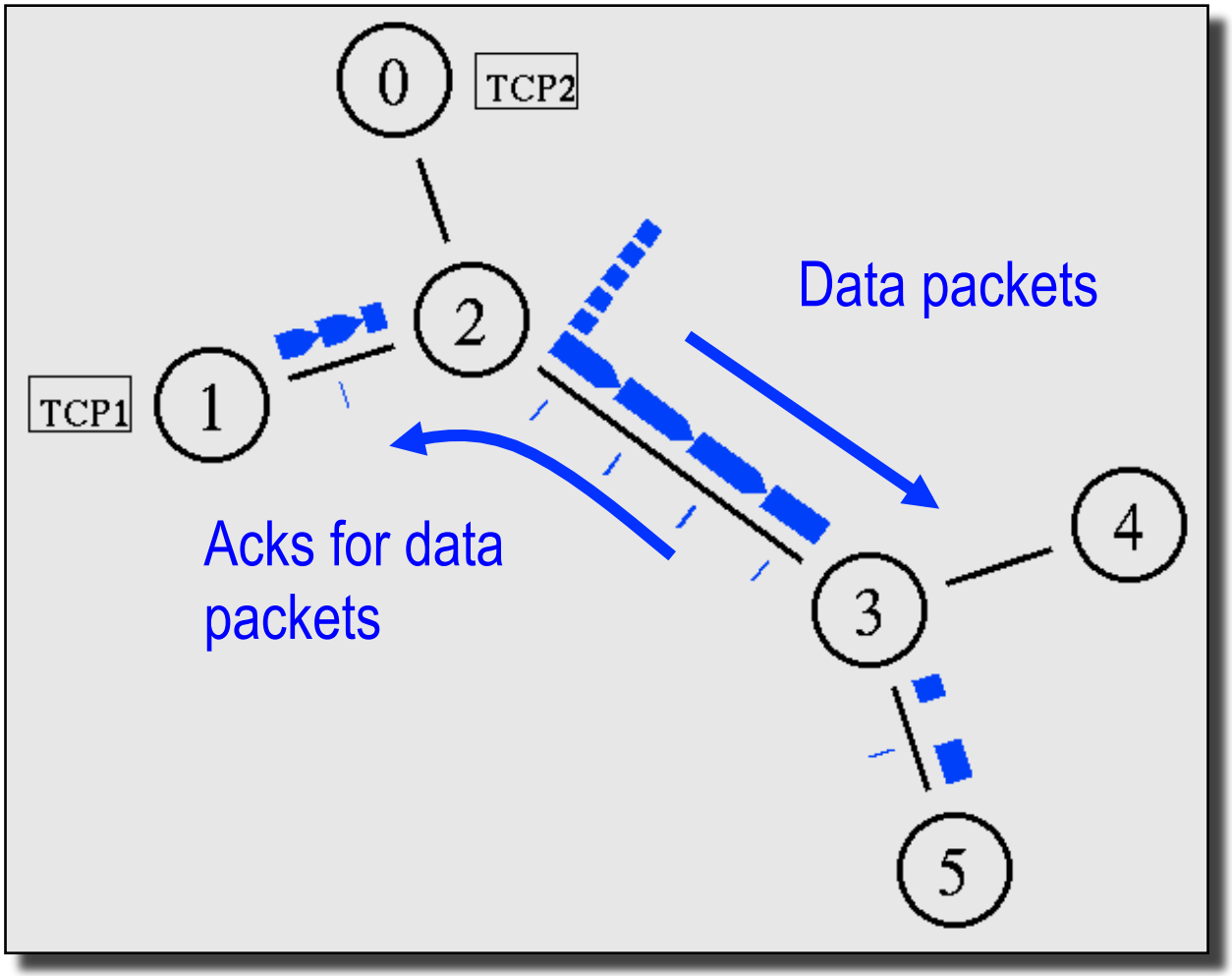
This is largely because most applications use TCP, and TCP implements *end-to-end congestion control.*

# Primary Goals of Congestion Control

(from a network point of view)

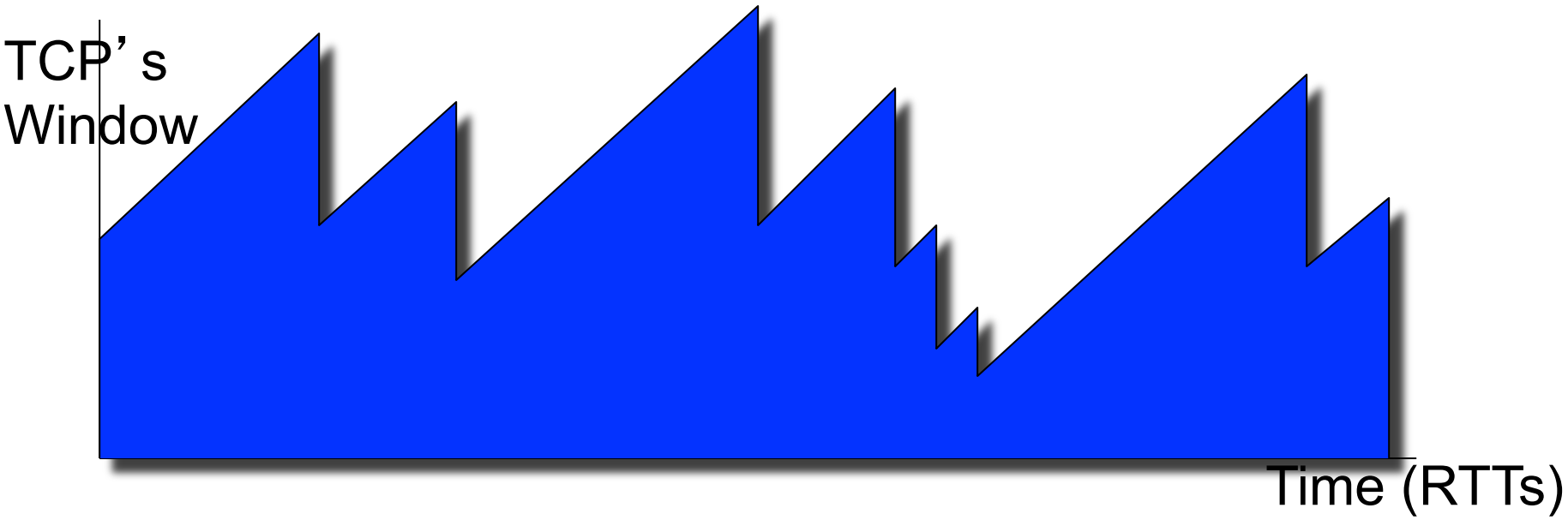
1. Avoid **congestion collapse**
   - Network must work.
2. Some sort of **fairness**
   - All users must get some service.

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

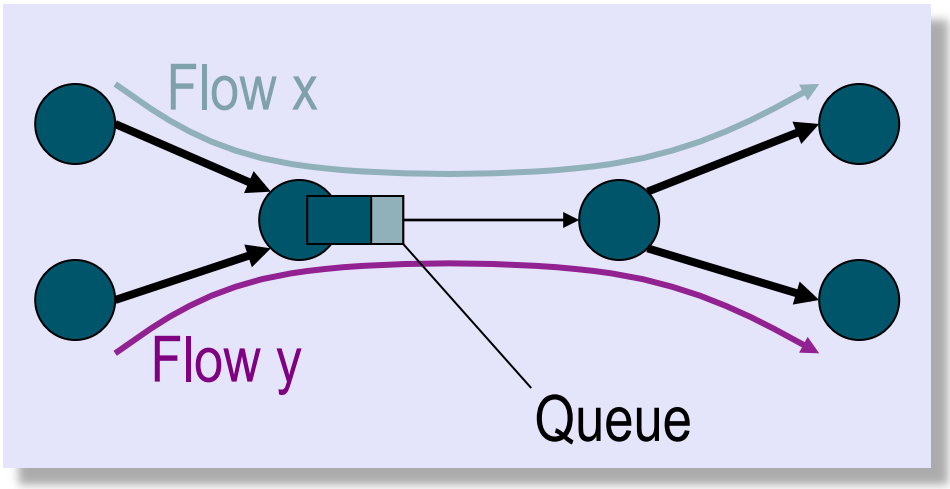


## TCP's congestion control adapts the window to fit the capacity available in the network.

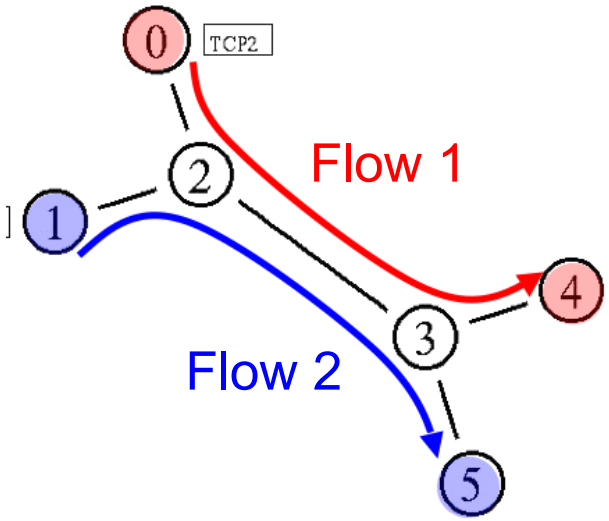
- Each round-trip time, increase window by one packet.
- If a packet is lost, halve the window.

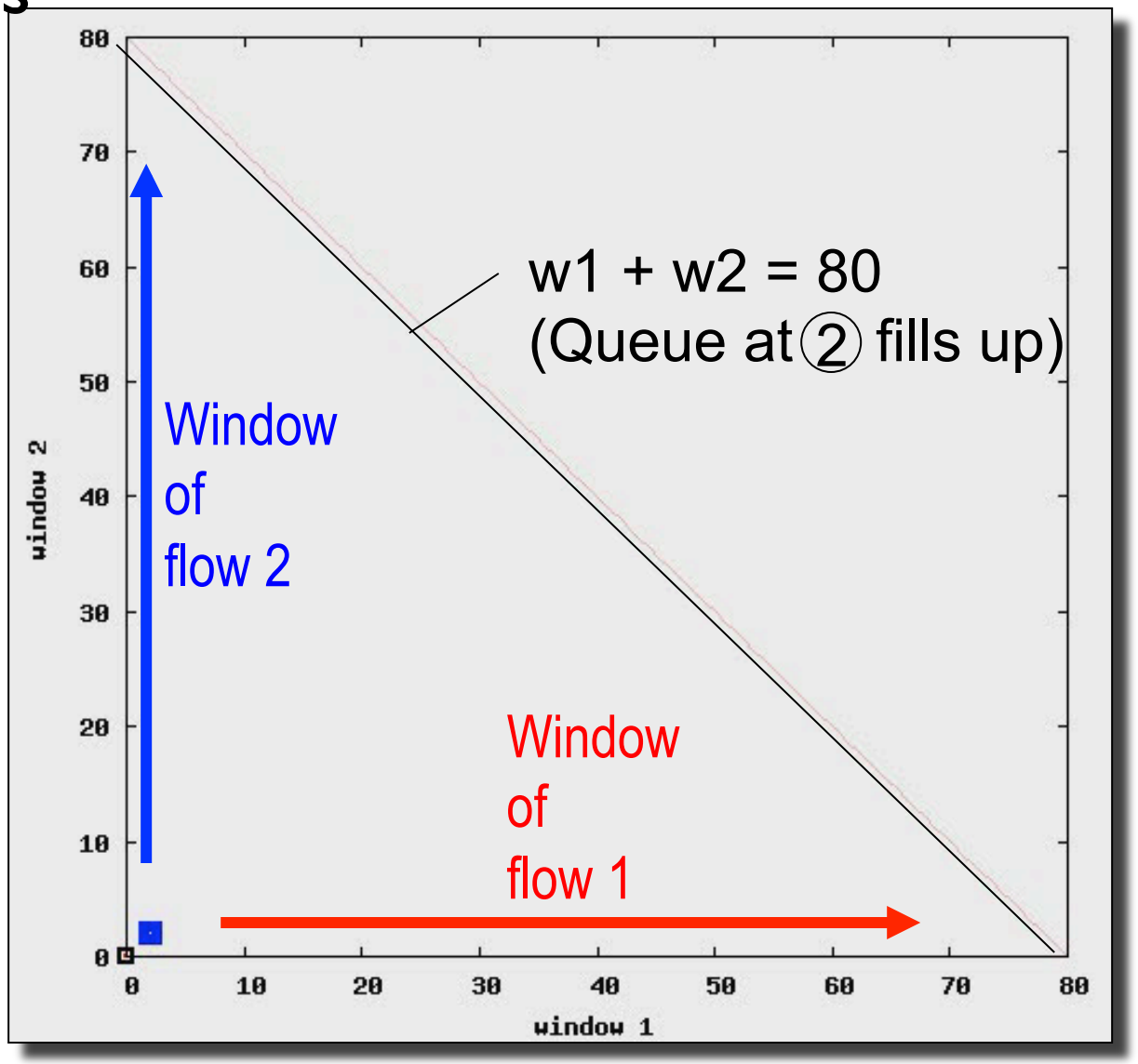


# TCP Fairness

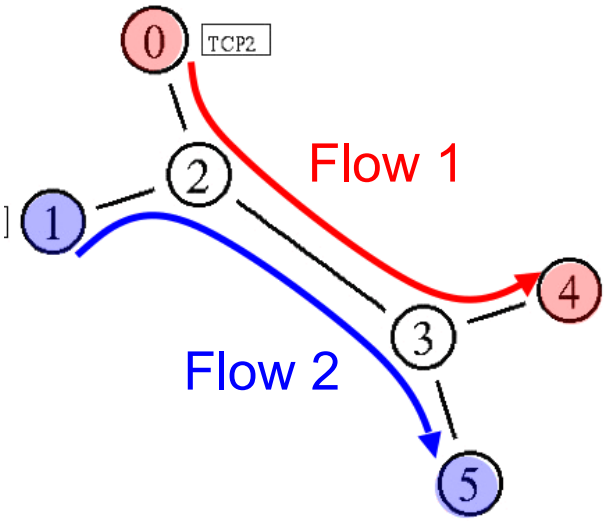


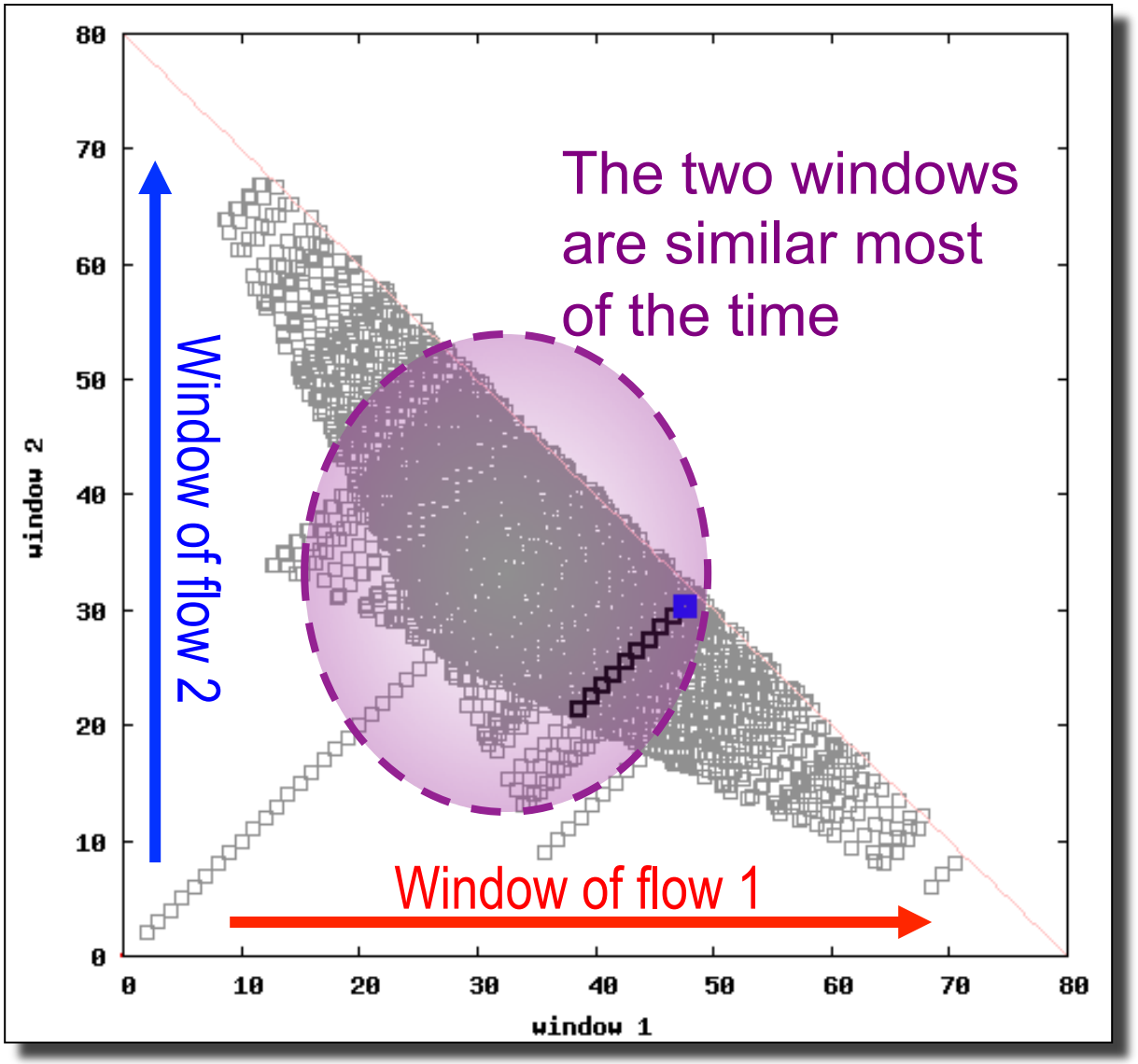
# Over time, TCP equalizes the windows of competing flows



# Over time, TCP equalizes the windows of competing flows

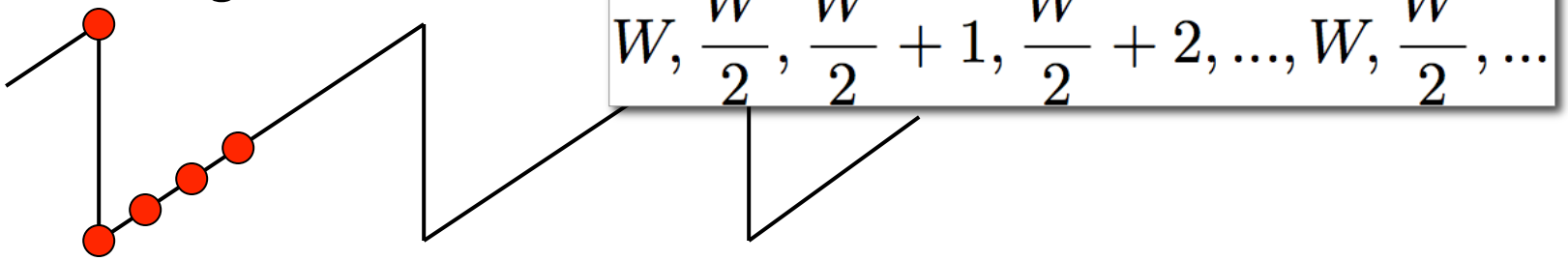


# TCP Modelling: The "Steady State" Model

**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 $p$: $p = \dfrac{1}{\frac{3}{8}W^2}$
- **The result:** $T = \dfrac{\sqrt{6}B}{2R\sqrt{p}} = \dfrac{\sqrt{3/2}B}{R\sqrt{p}}$

# Is TCP's fairness what we want?

- Probably not.
  - $1/R$ relationship not wonderful
    - But does mean you can reduce packet loss by adding big buffers!
  - $1/\mathrm{sqrt}(p)$ relationship not wonderful
    - Doesn't work well for high-speed flows.
- But at least it avoids starvation.

# What, precisely, are we controlling?

- TCP controls the congestion window in bytes.
  - For bulk transfer, usually this results in controlling the number of 1500 byte **packets per second** sent.
- Real-time media is different.
  - Audio:  one packet per (say) 20ms.
  - Video:  30 frames per second.
- Don't want to add extra delay.

# Video congestion control

- 30fps x 1500 bytes = 360Kbit/s
- Above this rate, we're sending more than one packet per frame (on average).
  - But no real precision on number of packets per frame til (say) > 1Mbit/s
- Below this rate, what to do?
  - Just send smaller packets at 30pps?

# What's the bottleneck?

- **Serial line:** bottleneck is in bits/second.
  - Doesn't matter how many packets/sec, so long as you include the packet headers in the calculation.

- **WiFi:**
  - At higher bitrates, MAC dominates.
  - Reducing the packet size makes little difference to available capacity.
  - Need to reduce packets/sec to relieve congestion.

# What's the bottleneck?

- Really do need to adapt packets/sec, not just bytes/sec.
- For video, with it's natural framerate, this clashes with what the application would prefer to do to get low latency.

# *Minimal* Primary Goals of Congestion Control
## (from a network point of view)

- Avoid congestion collapse
- Avoid starvation
  - Of TCP flows
  - Of real-time flows
  - Sharing same FIFO queue

# Goals of Congestion Control (from an application point of view)

- Robust behavior
- Predictable behavior
- Low latency

# Robust Behavior

- Good quality when network is working well.
- Still works when network is working poorly.
  - Loss is low enough for session still be be useful.

# Predictable Behavior

- Variable quality is bad for users.
- User studies:
  - When quality varies, rate overall quality close to minimum of qualities seen.

# Low Latency

- If you share a congested link with TCP, good luck to you.
  - Bufferbloat means you'll often get unwanted latency for your multimedia sharing the link.
- Delay based vs loss based congestion control.
  - Delay-based congestion control can keep latency low.
  - But if you're competing in the same queue as TCP, TCP will dictate the latency.

# Streaming vs Interactive Media

- Streaming:
  - Use buffering to smooth out throughput variation
  - Streaming over TCP is common.
  - Streaming over UDP is probably better, but…
- Interactive:
  - Can't afford to buffer (much) or latency will be unacceptable.

# TCP-Friendly Rate Control (TFRC)

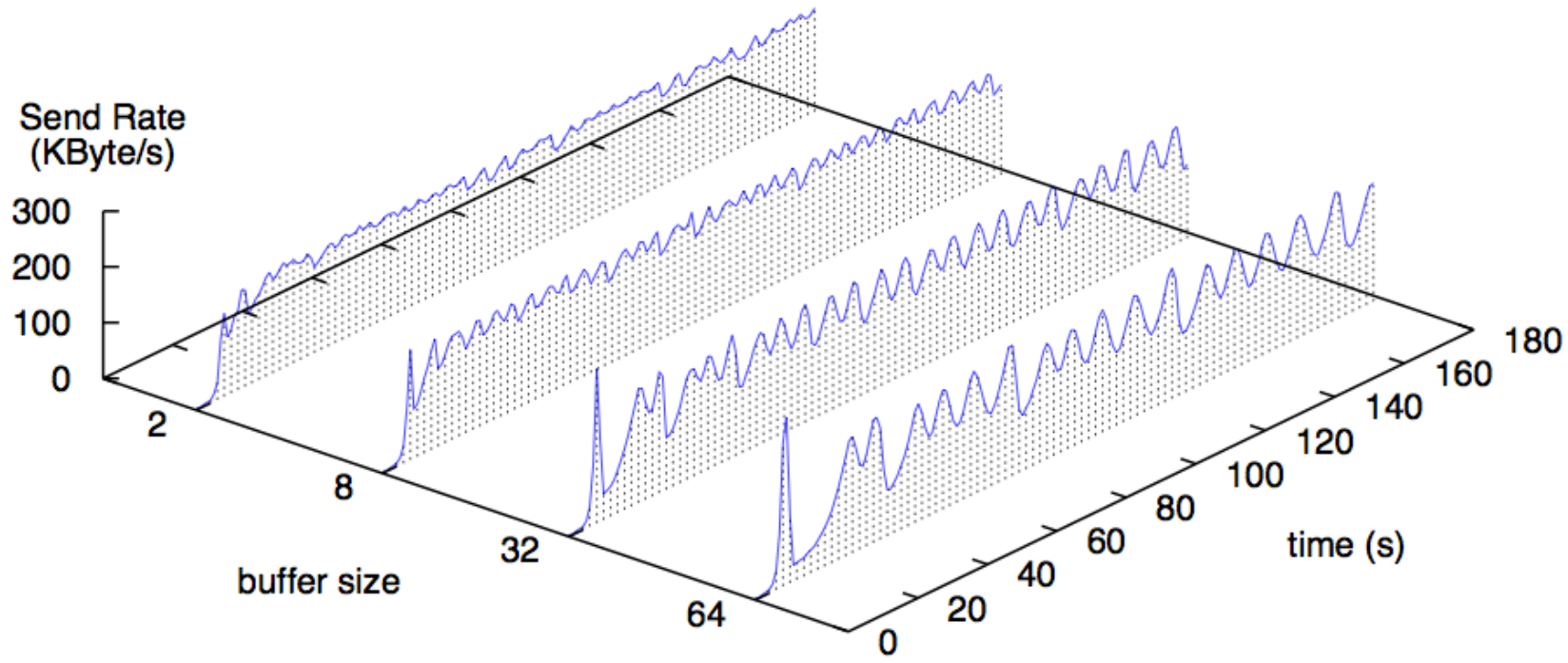
## RFC 5348

- Goal:
  - Same throughput as TCP, but smoother
- Mechanism:
  - Measure loss, RTT.
  - Use TCP model to determine sending rate.

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

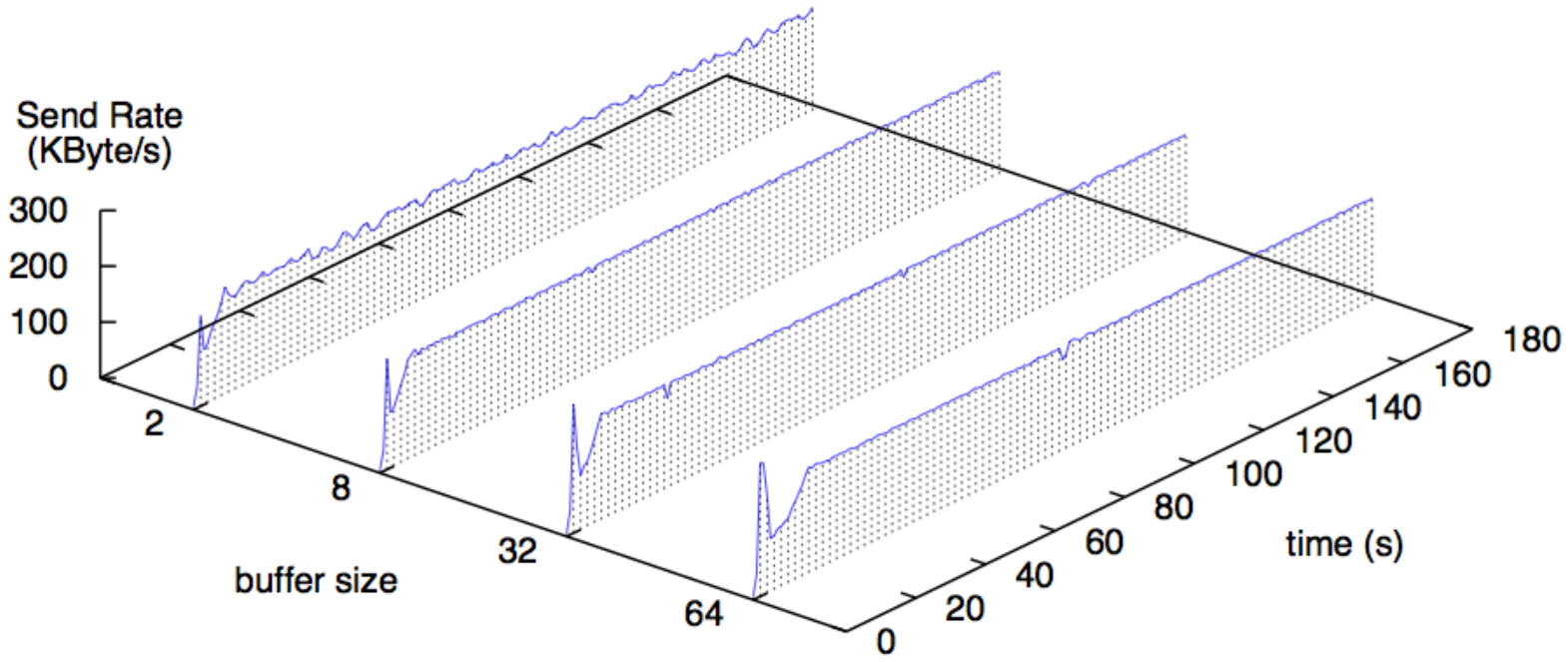
# TFRC Problem

- Oscillation with low statmux bottleneck
  - Eg. TFRC is only flow on a DSL link.



# TFRC Problem

- Oscillation with low statmux bottleneck
- Solution: short-term rate adaptation based on RTT.

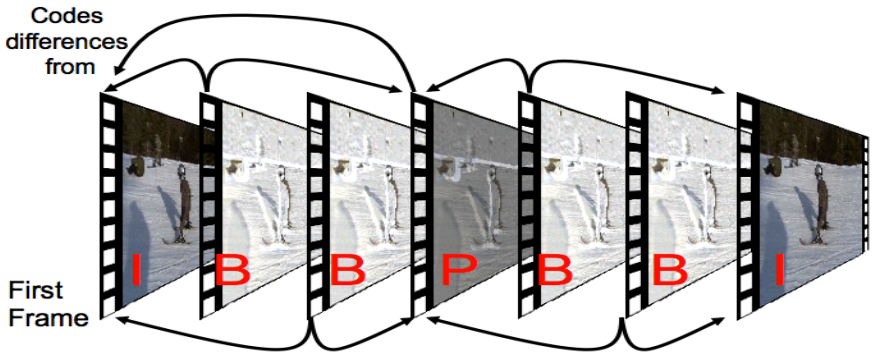


# Is this all the wrong approach?

- TFRC assumes network is in charge of codec.
  - Congestion control clocks out the packets into the network (just like TCP does).
- Assumes codec can produce data at demanded rate.

# Video is Inherently Variable

- Content sensitivity.
  - Motion, scene changes, etc
- MPEG 2:




I-frame: 50KB, P-frame: 25KB, B-frame: 10KB.

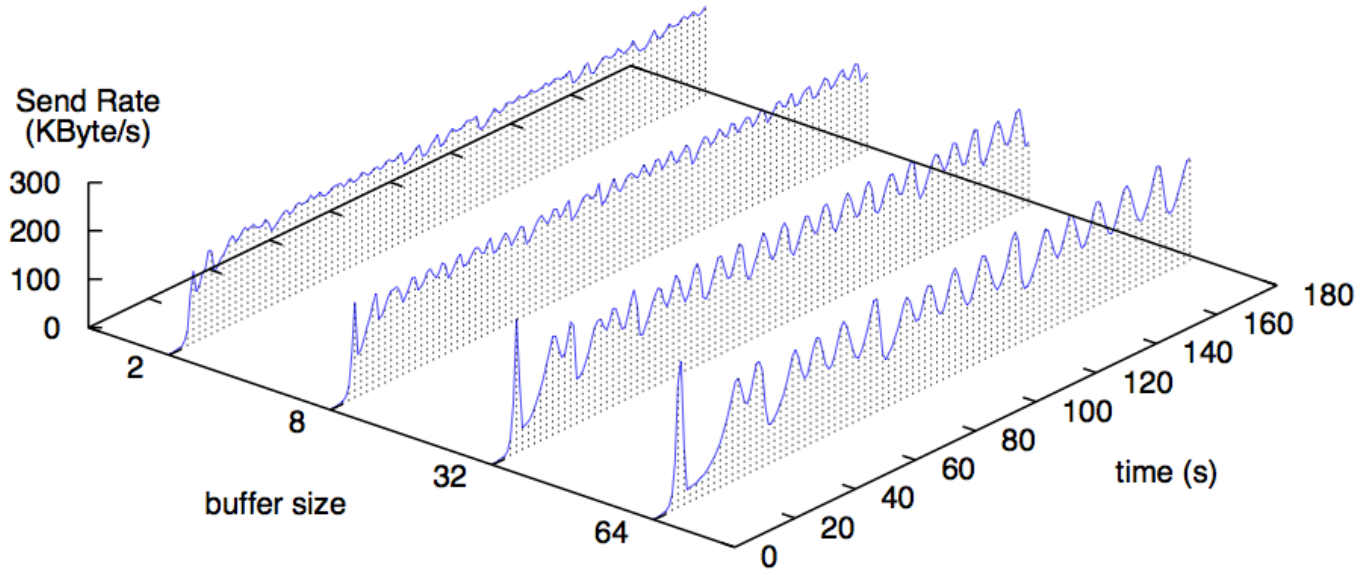
- Hard to produce data at exactly a demanded rate without excessive buffering or ugly quality changes.

# What if codec is in charge?

- Net tells codec mean rate to send.
- Codec matches mean rate, but not on short timescales.

# The low statmux problem

- Known congestion control mechanisms work well if they can respond fast to changes.
  - If not, can bump into linkspeed cap, get very high loss, and then overreact. Result: Oscillation



# Linkspeed characterization

- Techniques for inferring linkspeed of lowest speed link
  - *pathchar* and its successors
  - Send short packet trains.
  - Measure timing accurately.
  - Can infer linkspeed from relative delay.

- If you know the linkspeed, you can avoid exceeding it, even for short durations.

# Link unpredictability

- WiFi
  - fading
  - link-local RTX
  - WiFi bitrate adaptation
- Fair queuing.
  - Eg WFQ in home gateways.

# RTP

- RTCP feedback:
  - What can be sent?
    - Loss rate, RTT need measuring.
    - RTP XR allows more.
  - How often?
    - Not often enough for fine-grain RTT measurement

- We wrote these standards. No need to be limited, *if we know what we want*.

# Circuit Breakers

- Measure loss-rate, RTT.
  - Calculate TCP rate.
  - Stop sending if application rate is too high for too long.

- Avoid congestion collapse.  Good!
- But miss out on potential benefits of good congestion control.

# Summary

- We know how to do reasonable congestion control.
- But only if CC is in charge.
  - Probably not acceptable
  - Only if we change pps rate, not just packet size.

- This is a multifaceted problem – need to balance application requirements with reasonable network behaviour.
  - Not (yet) a solved problem.