# Congestion control - how to cope with traffic jams on the Internet

Michael Welzl http://www.welzl.at
Institute of Computer Science
University of Innsbruck, Austria

# Control *what*? Traffic jams, huh?

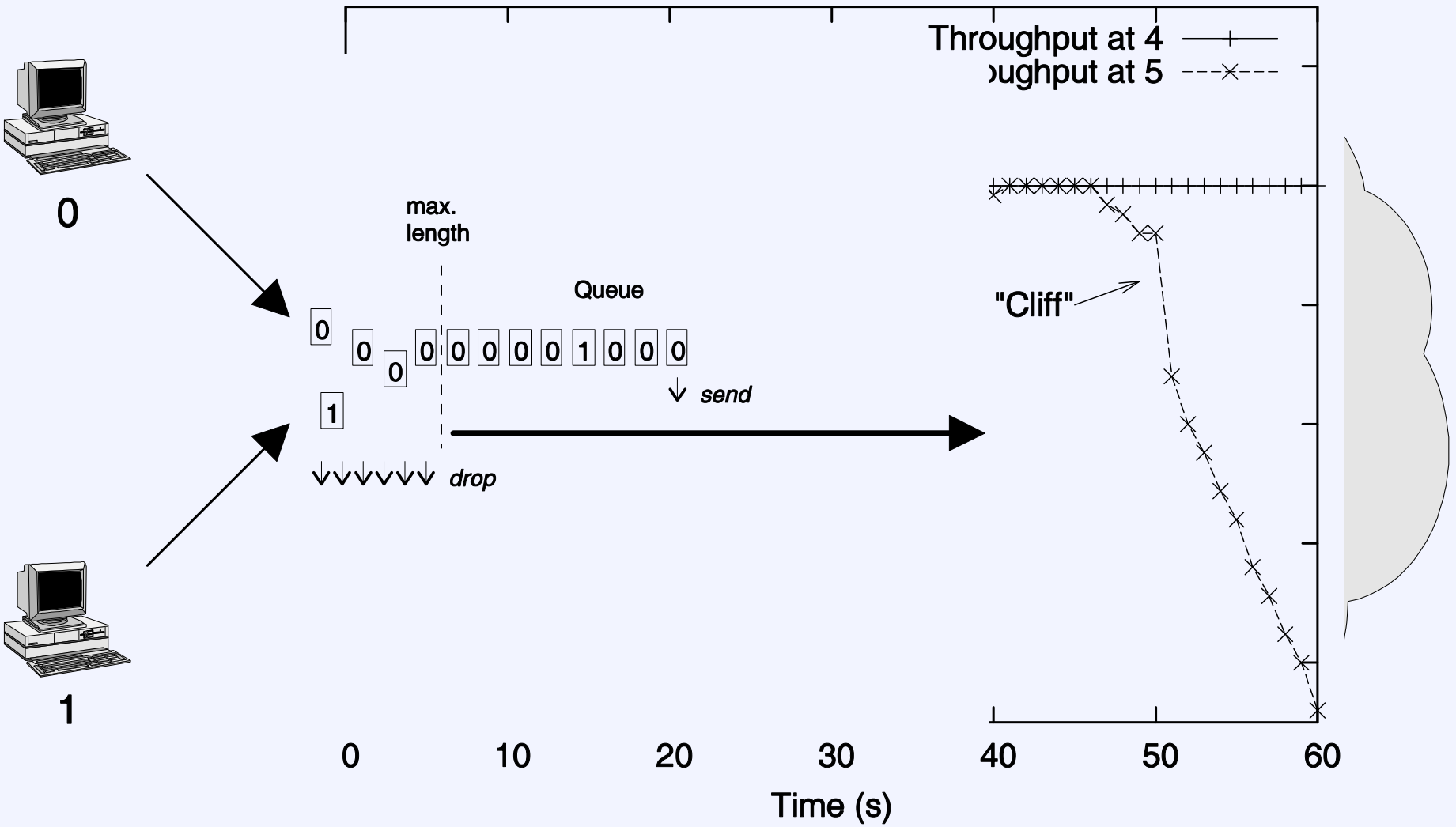
- Nowadays, networks are often overprovisioned
  $\Rightarrow$ no traffic jams; no congestion
  - often, but not always (e.g. wireless links)
  - this situation may change (access vs. core bandwidth changes)

- Networks are underutilized...
  exactly, that's the issue!

- Essentially, the problem changed from „*how do we get rid of all this congestion*" to „*how do we efficiently use all this spare bandwidth*"

- **Outline**
  - Introducing Internet congestion control
  - Problems, proposed solutions

# Congestion collapse



# Global congestion collapse in the Internet

Craig Partridge, Research Director for the Internet Research Department at BBN Technologies:

Bits of the network would fade in and out, but usually only for TCP. You could ping. You could get a UDP packet through. Telnet and FTP would fail after a while. And it depended on where you were going (some hosts were just fine, others flaky) and time of day (I did a lot of work on weekends in the late 1980s and the network was wonderfully free then).

Around 1pm was bad (I was on the East Coast of the US and you could tell when those pesky folks on the West Coast decided to start work...).

Another experience was that things broke in unexpected ways - we spent a lot of time making sure applications were bullet-proof against failures. (..)

Finally, I remember being startled when Van Jacobson first described how truly awful network performance was in parts of the Berkeley campus. It was far worse than I was generally seeing. In some sense, I felt we were lucky that the really bad stuff hit just where Van was there to see it.

# Internet congestion control: History

- **1968/69:** dawn of the Internet
- **1986:** first congestion collapse
- **1988:** "Congestion Avoidance and Control" (Jacobson)
  Combined congestion/flow control for TCP
  (also: variation change to RTO calculation algorithm)

- Goal: stability - in equilibrum, no packet is sent into the network until an old packet leaves
  - ack clocking, "conservation of packets" principle
  - made possible through window based stop+go - behaviour

- Superposition of stable systems = stable →
  network based on TCP with congestion control = stable

# TCP Congestion Control: Tahoe

- Distinguish:
  - flow control: protect receiver against overload
    (receiver "grants" a certain amount of data ("receiver window" (rwnd)) )
  - congestion control: protect network against overload
    ("congestion window" (cwnd) limits the rate: min(cwnd,rwnd) used! )

- Flow/Congestion Control combined in TCP. Two basic algorithms:
  (window unit: SMSS = Sender Maximum Segment Size, usually adjusted to Path MTU;
  init cwnd<=2 (*SMSS), ssthresh = usually 64k)

- Slow Start: for each ack received, cwnd += 1 until >= ssthresh (exponential growth)

- Congestion Avoidance: each RTT, increase cwnd by SMSS*SMSS/cwnd
  (linear growth - "additive increase")
  - common implementation: update cwnd like this for every ACK in this mode.
    → wrong behaviour with delayed ACKs; solution: Appropriate Byte Counting (ABC)

- Timeout: ssthresh = FlightSize/2 (exponential backoff - "multiplicative decrease"),
  cwnd = 1; FlightSize = packets in flight (may be less than cwnd)

# Fast Retransmit / Fast Recovery (Reno)

From RFC 2581; underlying idea: cwnd = number of packets in flight

1. Upon reception of third duplicate ACK (DupACK): ssthresh = FlightSize/2

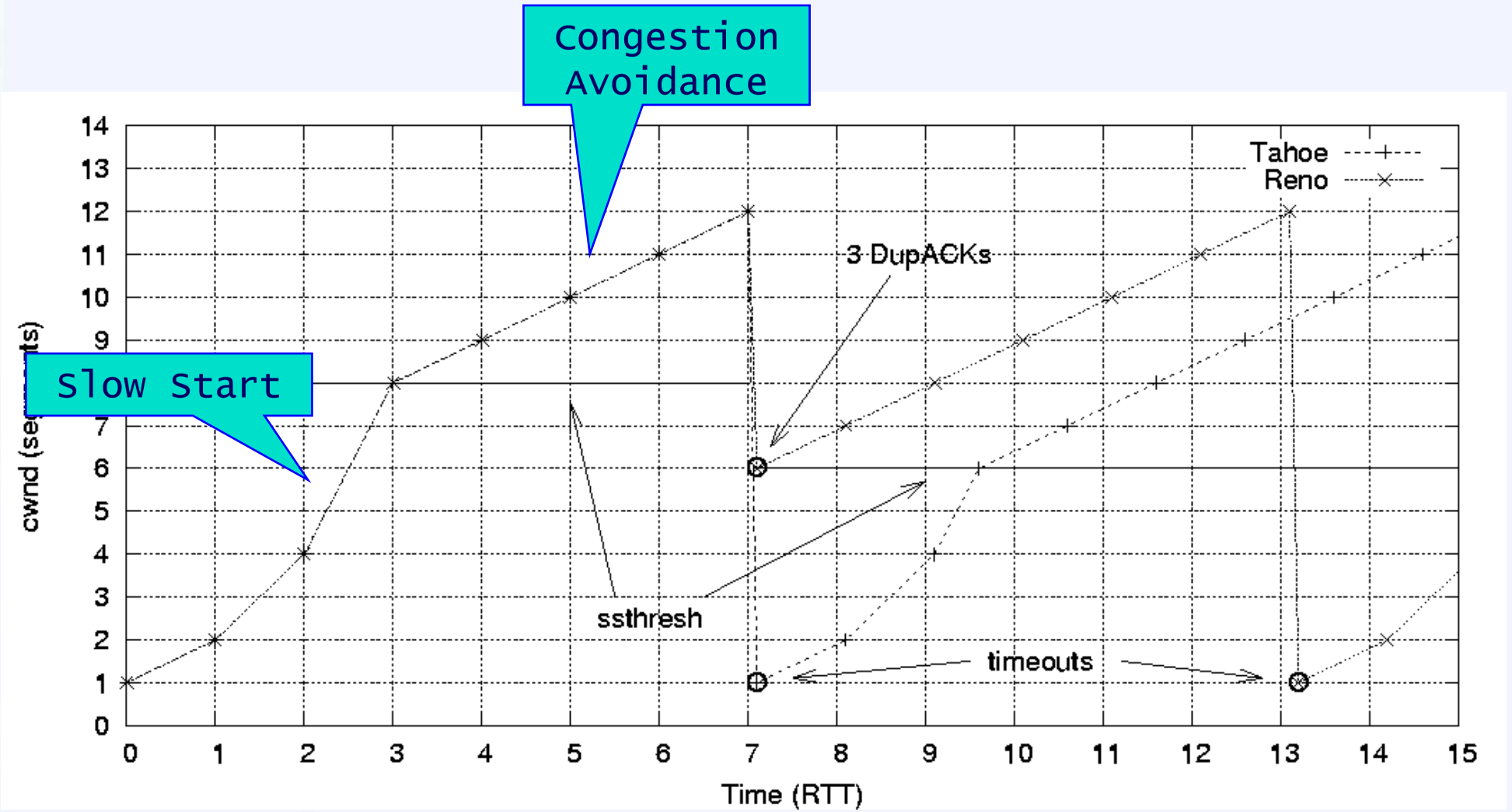
2. Retransmit lost segment (fast retransmit);
   cwnd = ssthresh + 3*SMSS
   ("inflates" cwnd by the number of segments (three) that have left the network and which the receiver has buffered)

3. For each additional DupACK received: cwnd += SMSS
   (inflates cwnd to reflect the additional segment that has left the network)

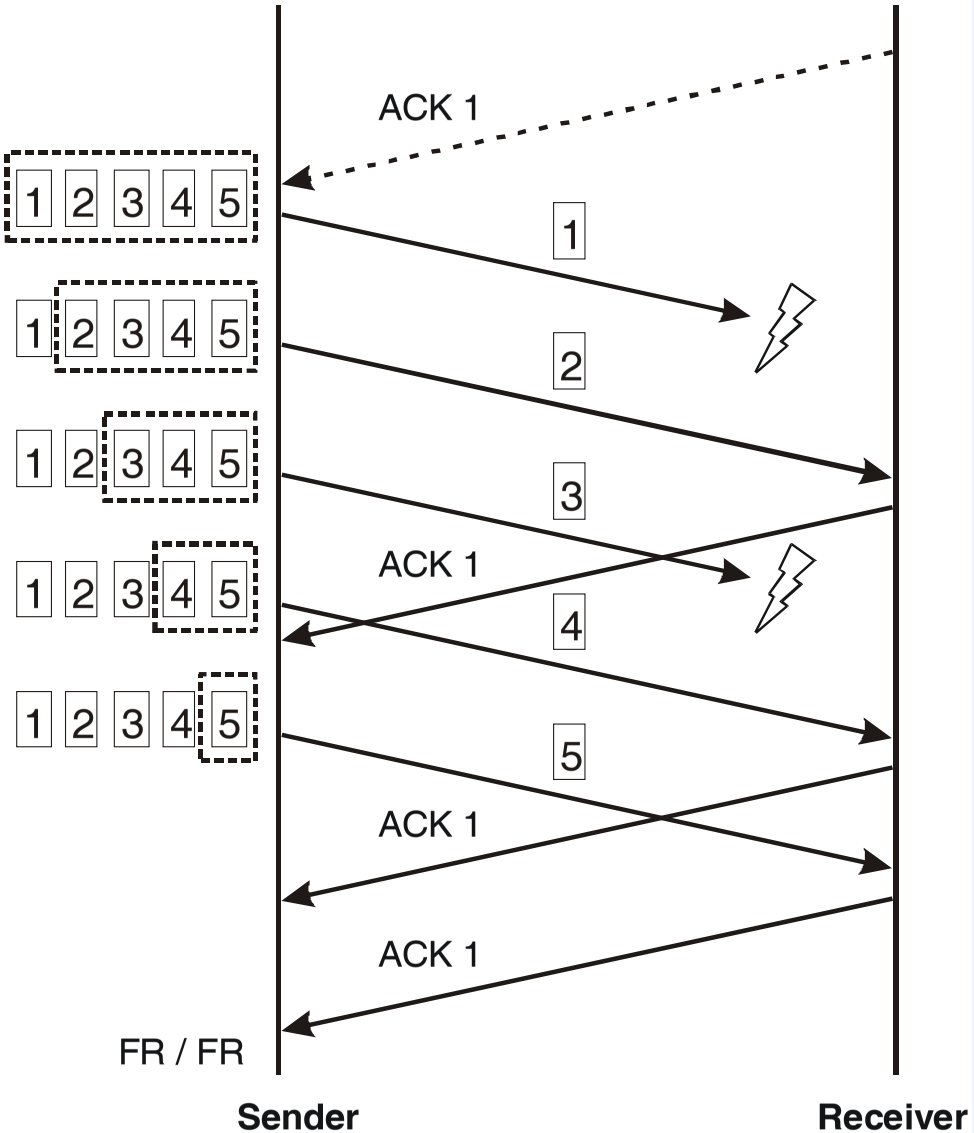
4. Transmit a segment, if allowed by the new value of cwnd and rwnd

5. Upon reception of ACK that acknowledges new data ("full ACK"):
   "deflate" window: cwnd = ssthresh (the value set in step 1)

# Tahoe vs. Reno



# One window, multiple dropped segments



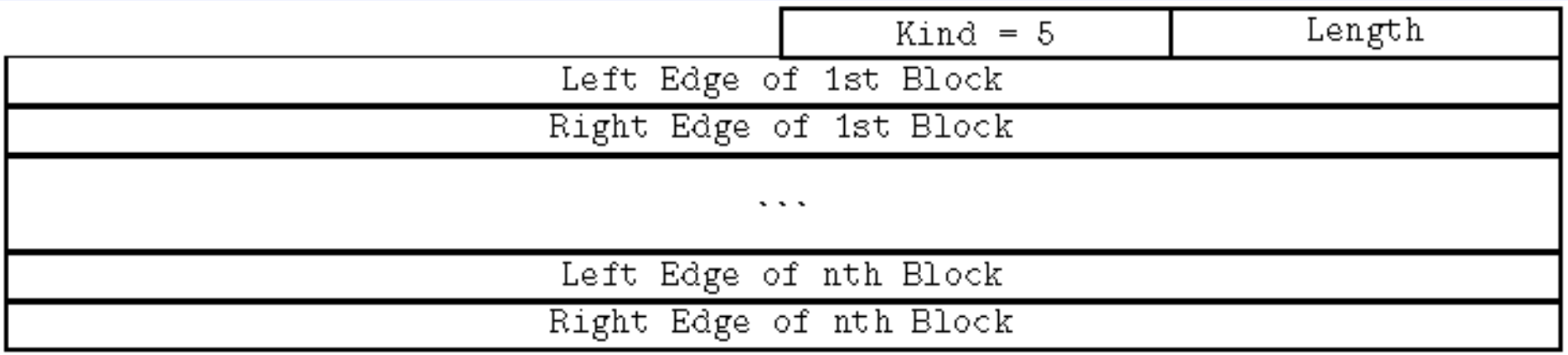

- Sender cannot detect loss of multiple segments from a single window
- Insufficient information in DupACKs

Example: ACK 2

- NewReno:
  - stay in FR/FR when partial ACK arrives after DupACKs
  - retransmit single segment
  - only full ACK ends process

Example: ACK 6

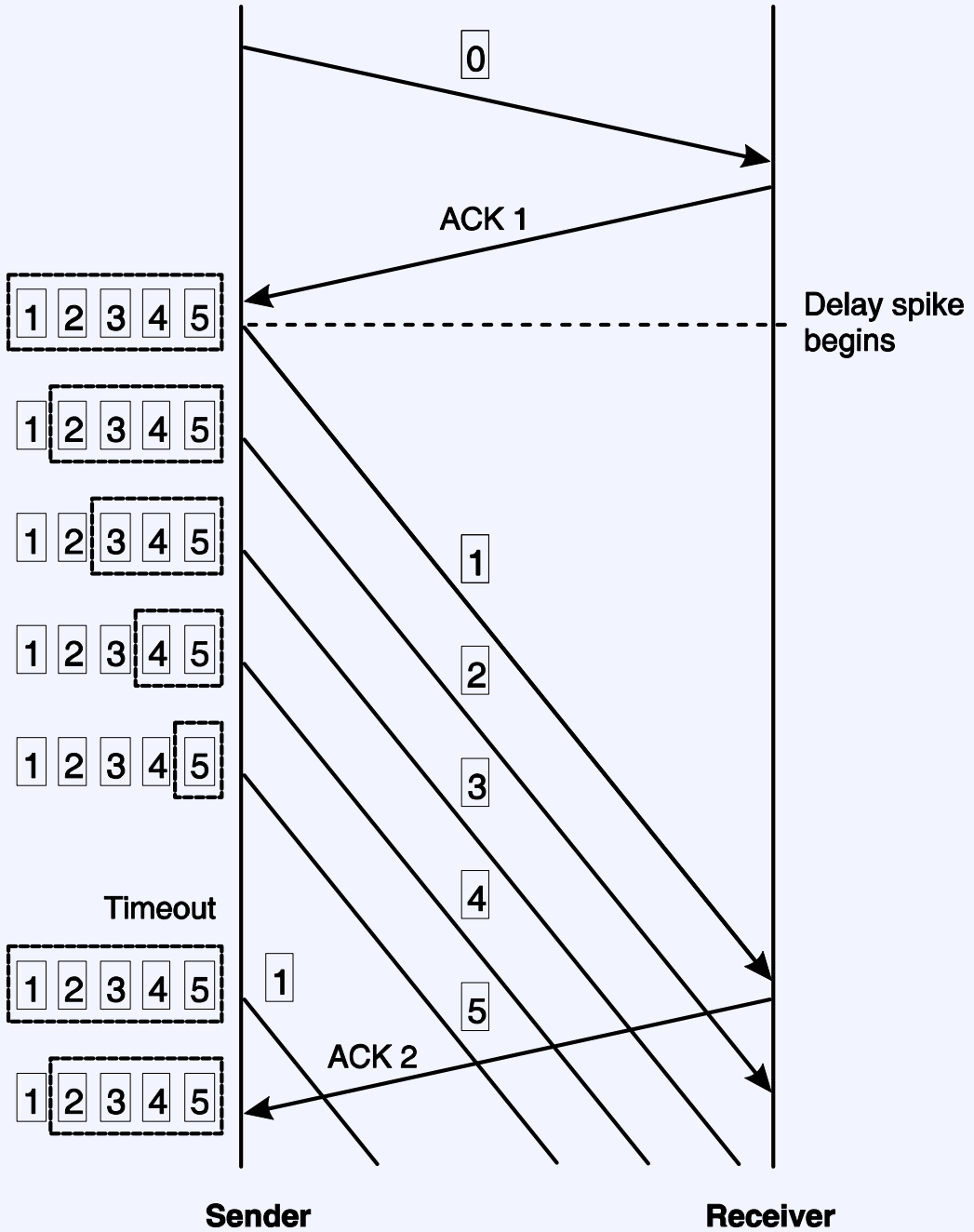
# Selective ACKnowledgements (SACK)




- Example on previous slide:
  send ACK 1, SACK 3, SACK 5 in response to segment #4
- Better sender reaction possible
  - Reno and NewReno can only retransmit a single segment per window
  - SACK can retransmit more (RFC 3517)
  - Particularly advantageous when window is large (long fat pipes)
- but: requires receiver code change

# Spurious timeouts

- Common occurrence in wireless scenarios (handover): sudden delay spike
- Can lead to timeout
  → slow start
  - But: underlying assumption: "pipe empty" is wrong! ("spurious timeout")
  - Old incoming ACK after timeout should be used to undo the error
- Several methods proposed; e.g. Eifel Algorithm: use timestamps option to check: timestamp in ACK < time of timeout?



# Active Queue Management

- Today, TCP behaviour dominates the Internet (WWW, ..)
- (somewhat old) example backbone measurement: 98% TCP traffic
- **1993:** Random Early Detection ("Discard", "Drop") (RED)
  (now that end nodes back off as packets are dropped,
  drop packets earlier to *avoid* queue overflows)
- Another goal: add randomization to avoid traffic phase effects!
- ***Qavg = (1 - Wq) x Qavg + Qinst x Wq***
  (Qavg = average occupancy, Qinst = instantaneous occupancy,
  Wq = weight - hard to tune, determines how aggressive RED behaves)

# Active Queue Management /2

- Based on exponentially weighted moving average (EWMA) of instantaneous queue occupancy = low pass filter
  - recalculated every time a packet arrives

- *Qavg* below threshold *min_th*: Nothing happens
- *Qavg* above threshold *min_th*: Drop probability rises linearly
- *Qavg* above threshold *max_th*: Drop packets

- RED expects all flows to behave like TCP - but is it fair?

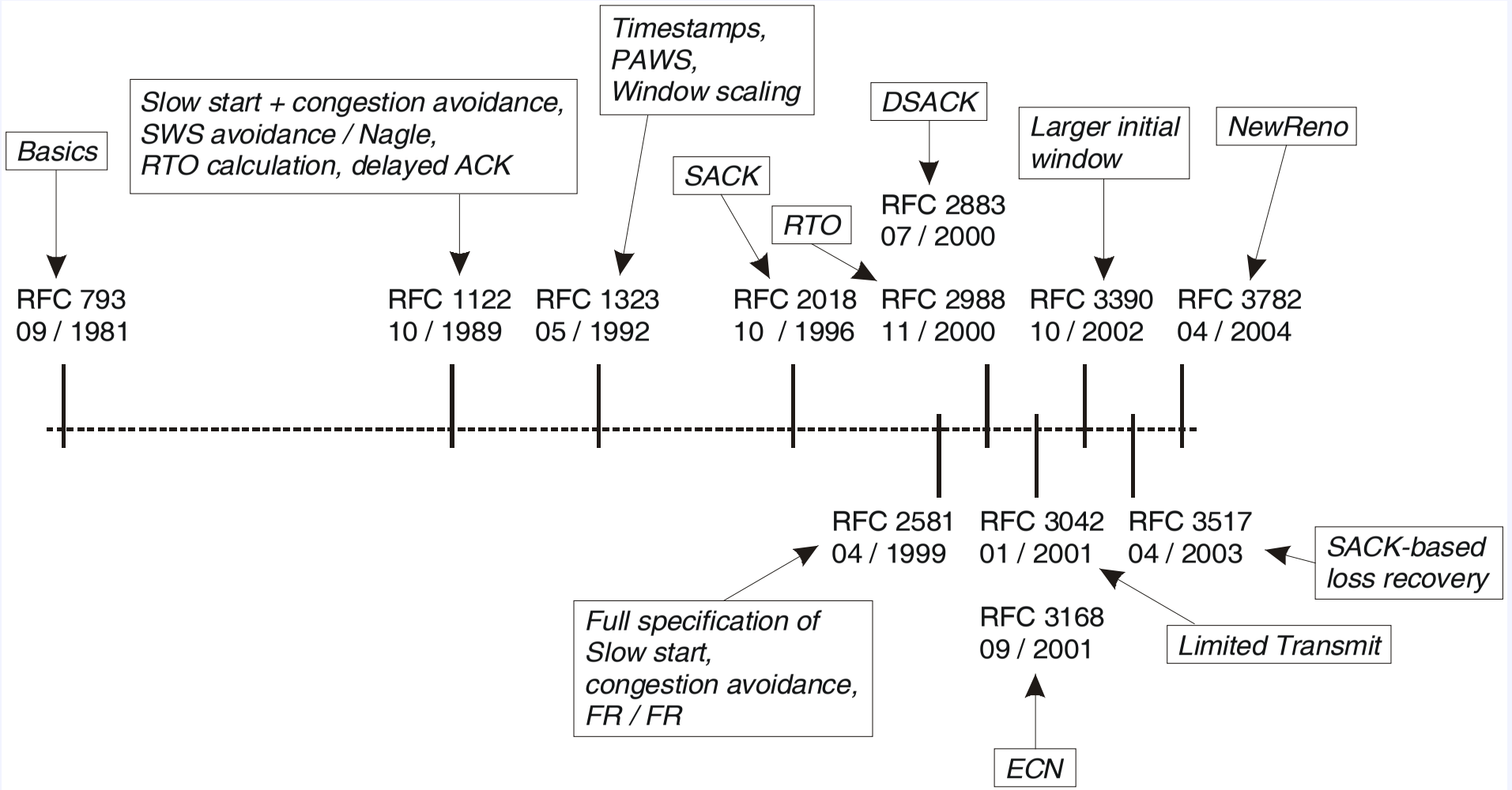
- Variants: drop from front, drop based on instantaneous queue occupancy, drop arbitrary packets, drop based on priorities...

# Explicit Congestion Notification (ECN)

- **1999:** Explicit Congestion Notification (ECN)
  Instead of dropping, set a bit
- End systems are expected to act as if packet was dropped
  $\Rightarrow$ actual communication between end nodes and the network!
- ATM and Frame Relay: not only ECN but also BECN
- Internet BECN: often proposed and regularly discussed (ICMP SQ), but very unlikely - several reasons
- Quite popular among researchers - lots of ideas to exploit the bit!
- ECN cannot totally replace loss measurements!

# TCP History

Standards track TCP RFCs which influence when a packet is sent



# Problems
# (and proposed solutions)

# End2end real-time data transfer

- Assumption: no special service available at application level
  - (Definition of Internet *"real-time"* softer than usual)

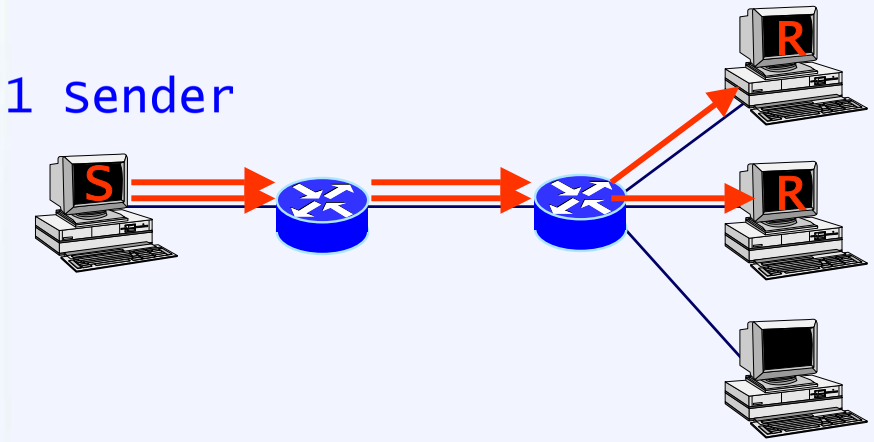
- Different requirements:
  - reliable service may not be needed (no retransmission)
  - Timely transmission important

- Different treatment:
  - no retransmission / waiting for ACKs
  - no sliding window (stop + go behaviour not suitable)
- but:
  - some kind of flow control still needed
  - synchronization necessary
  - often: Multicast

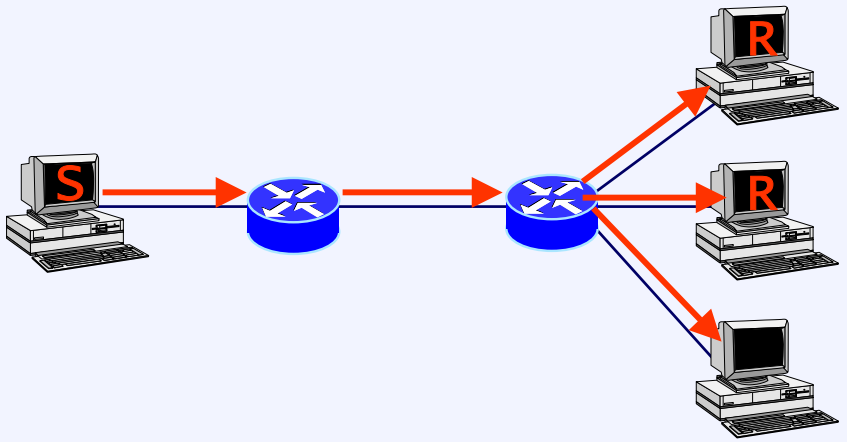
# Unicast / Broadcast / (overlay) Multicast

2 Receivers

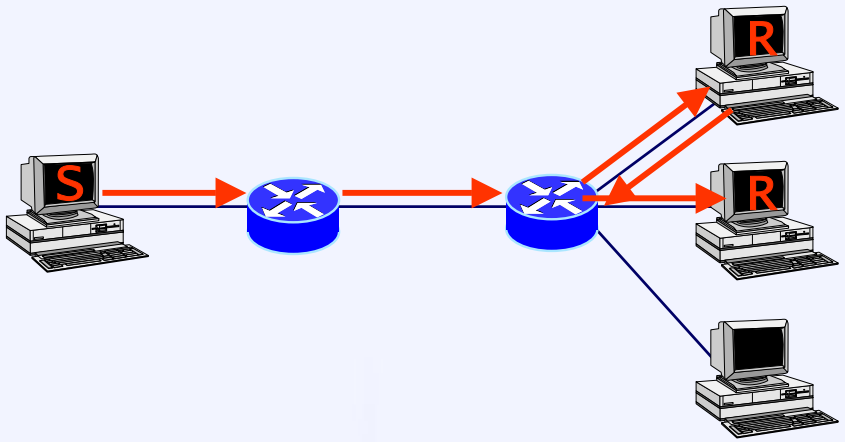
1 Sender



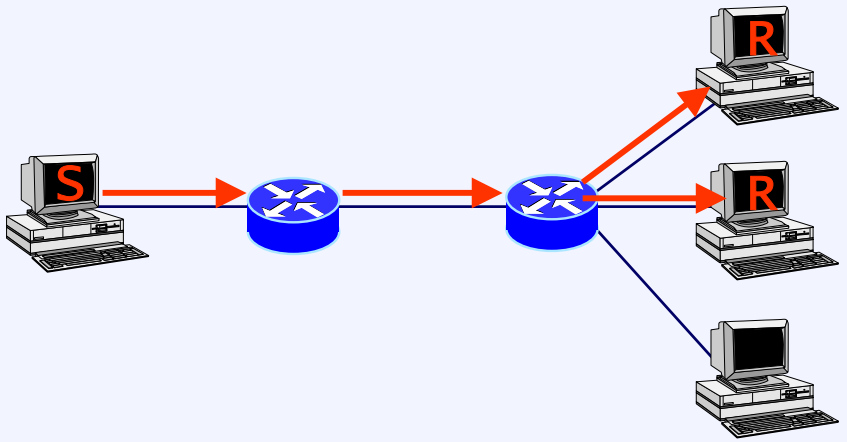

Unicast




Broadcast




Overlay Multicast




IP Multicast

# Multicast issues

- Required for applications with multiple receivers only
  - video conferences, real-time data stream transmission, ..
    ⇒ different data streams than web surfing, ftp downloads etc!

- Issues:
  - group management
    - protocol required to join / leave group dynamically:
      Internet Group Management Protocol (IGMP)
    - state in routers: hard / soft (lost unless refreshed)?
    - who initiates / controls group membership?
  - congestion control
    - scalability (ACK implosion)
    - dealing with heterogeneity of receiver groups
    - fairness

depends on content!

- Multicast congestion control mechanism classification:
  - sender- vs. receiver-based, single-rate vs. multi-rate (layered),
  - reliable vs. unreliable, end-to-end vs. network-supported

# Fairness

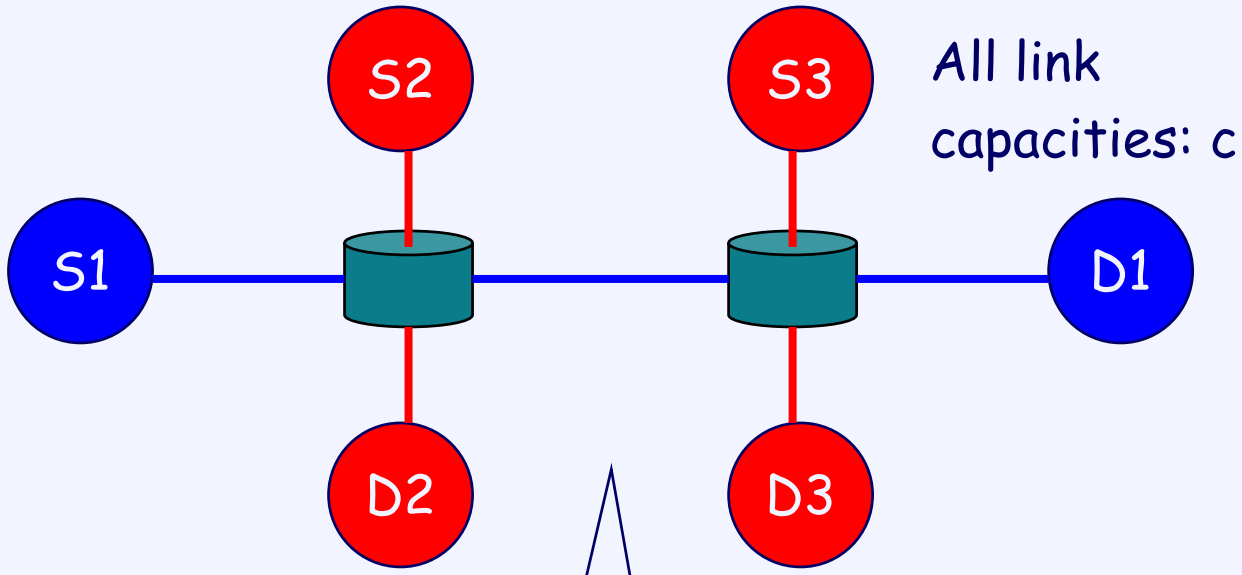
- ATM ABR: Max-Min-fairness
  - "A (..) allocation of rates is max-min fair iff an increase of any rate (..) must be at the cost of a decrease of some already smaller rate."
  - One resource: mathematical definition satisfies "general" understanding of fairness - resource is divided equally among competitors
  - *Usually* requires knowledge of flows in routers (switches) - scalability problem!

- Internet:
  - TCP dominant, but does not satisfy Max-Min-fairness criterion!
  - Ack-clocked - flows with shorter RTT react sooner (slow start, ..) and achieve better results
  - Therefore, Internet definition of fairness: TCP-friendliness

    "A flow is TCP-compatible (TCP-friendly) if, in steady state, it uses no more bandwidth than a conformant TCP running under comparable conditions."

# Proportional Fairness

F. Kelly: Network should solve a global optimization problem (maximize log utility function)




Max-Min-fairness suboptimal:
S1 = S2 = S3 = c/2

Proportional fairness:

"An allocation of rates $\vec{x}$ is proportionally fair iff, for any other (..) allocation $\vec{y}$, we have:

$$\sum_{s=1}^{S} \frac{y_s - x_s}{x_s} <= 0$$

"

*(**roughly** approximated by AIMD!)*

Proportionally fair allocation:
**S1 = c/3, S2 = S3 = 2c/3**

# Issues with TCP-friendliness

- TCP regularly increases the queue length and causes loss
  ⇒ detect congestion when it is already (ECN: almost) too late!
  - possible to have more throughput with smaller queues and less loss
    ... but: exceed rate of TCP under similar conditions ⇒ not TCP-friendly!

- What if I send more than TCP *in the absence* of competing TCP's?
  - can such a mechanism exist?
  - yes! TCP itself, with max. window size = bandwidth * RTT
  - Does this mean that TCP is not TCP-friendly?

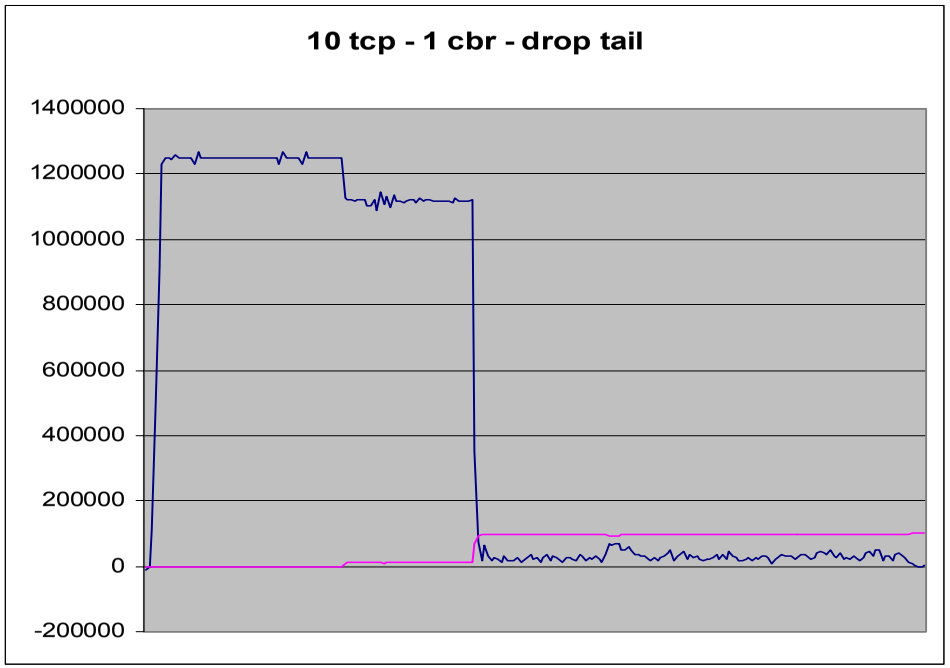
- Details missing from the definition:
  - parameters + version of „conformant TCP"
  - duration! short TCP flows are different than long ones

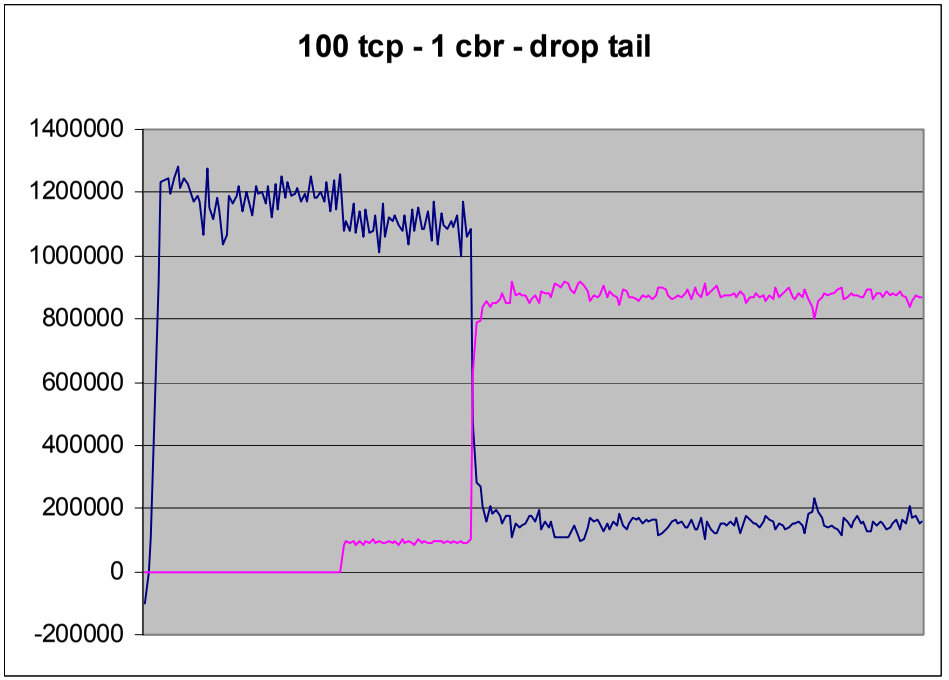
Does TCP-friendliness hinder research?

- TCP-friendliness = compatibility of new mechanisms with old mechanism
  - there was research since the 80's! e.g. new knowledge about network measurements

- TCP rate depends on RTT - how does this relate to „fairness"?

# On the other hand...




- For more details, see:
  *Promoting the Use of End-to-End Congestion Control in the Internet.*
  *Floyd, S., and Fall, K..*
  *IEEE/ACM Transactions on Networking, August 1999.*

# How to be TCP-friendly

- TCP-friendliness can be achieved by emulating the behaviour of TCP (or the desired parts of it)
- Simplified TCP: AIMD (additive incr. $\alpha$ , multiplicative decr. $\beta$)
  - **$0 < \alpha$ , $0 < \beta < 1$** -> stable and fair congestion control
  - **$\alpha = 4 \times (1 - \beta^2) / 3$** -> TCP-friendly congestion control (GAIMD)
  - **$\alpha = 1$, $\beta = 1/2$** -> TCP
- AIMD mechanisms for multimedia applications: RAP, LDA+
- Different approaches:
  - TCP Emulation At Receivers (TEAR)
    TCP calculations (cwnd calculation, fast recovery, ...) moved to receiver, do not ack every packet, smooth sending rate
  - Binomial congestion control: generalization of GAIMD with nonlinear control
  - CYRF framework: generalization of binomial congestion control

# Equation based congestion control

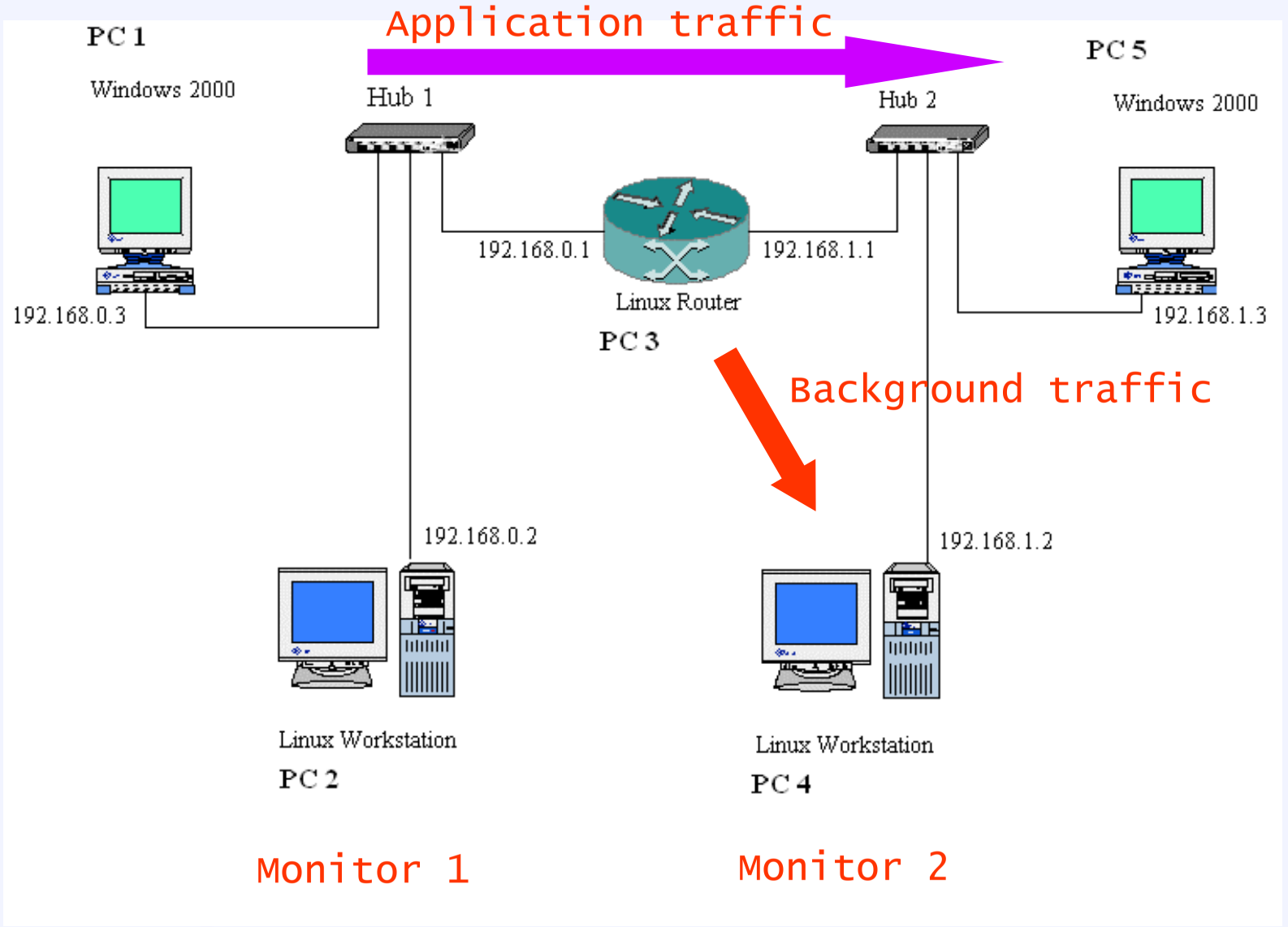
- Based on TCP steady-state response function
  - gives upper bound for transmission rate T (bytes/sec):

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

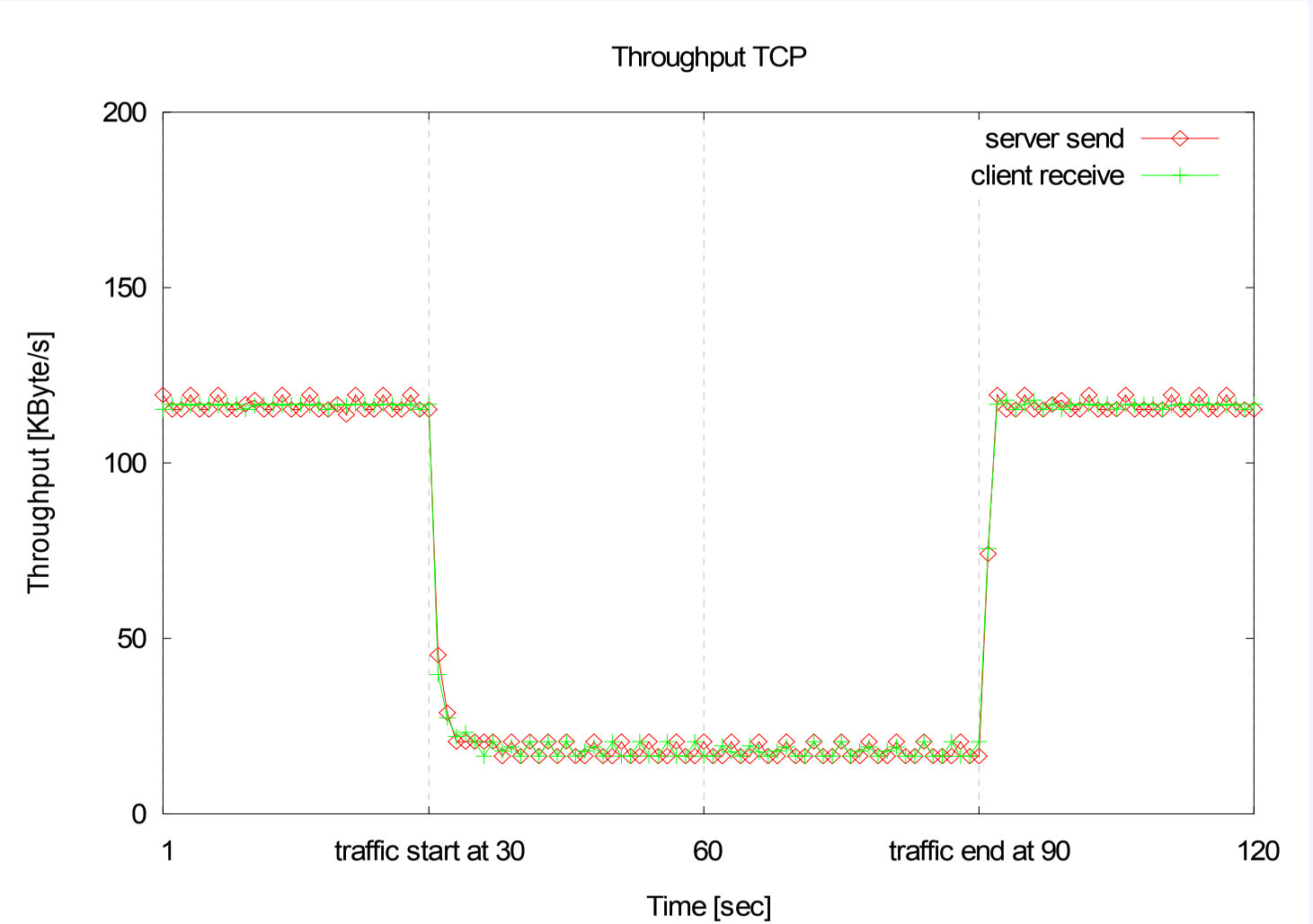
s: packet size
R: rtt
$t_{RTO}$: TCP retransmit timeout
p: steady-state loss event rate (the difficult part!)

- well known example: **TFRC** - TCP-friendly rate control protocol
  - smooth sending rate
- Extension: TFMCC - TCP-friendly multicast congestion control

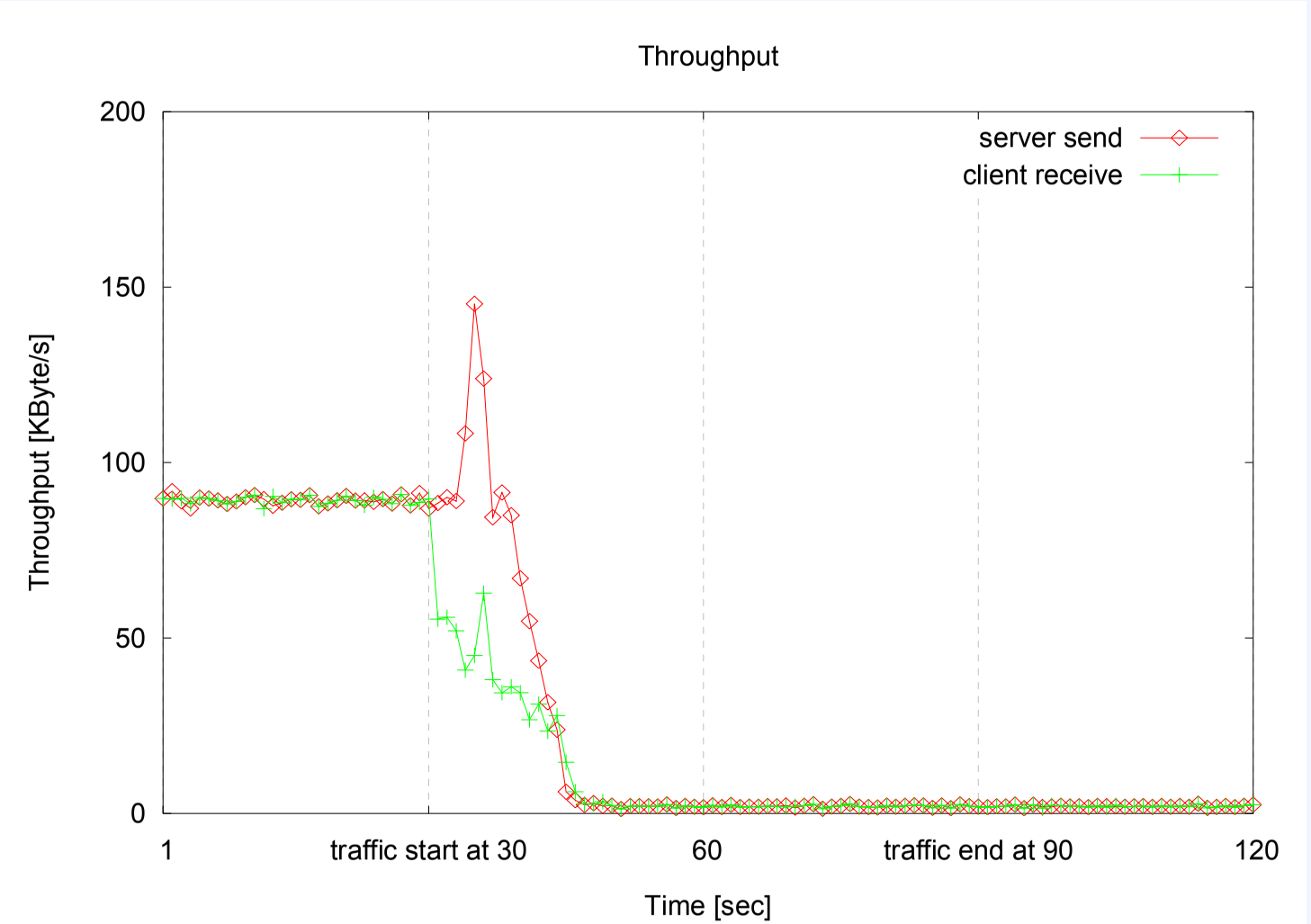
# Real behavior of today's apps



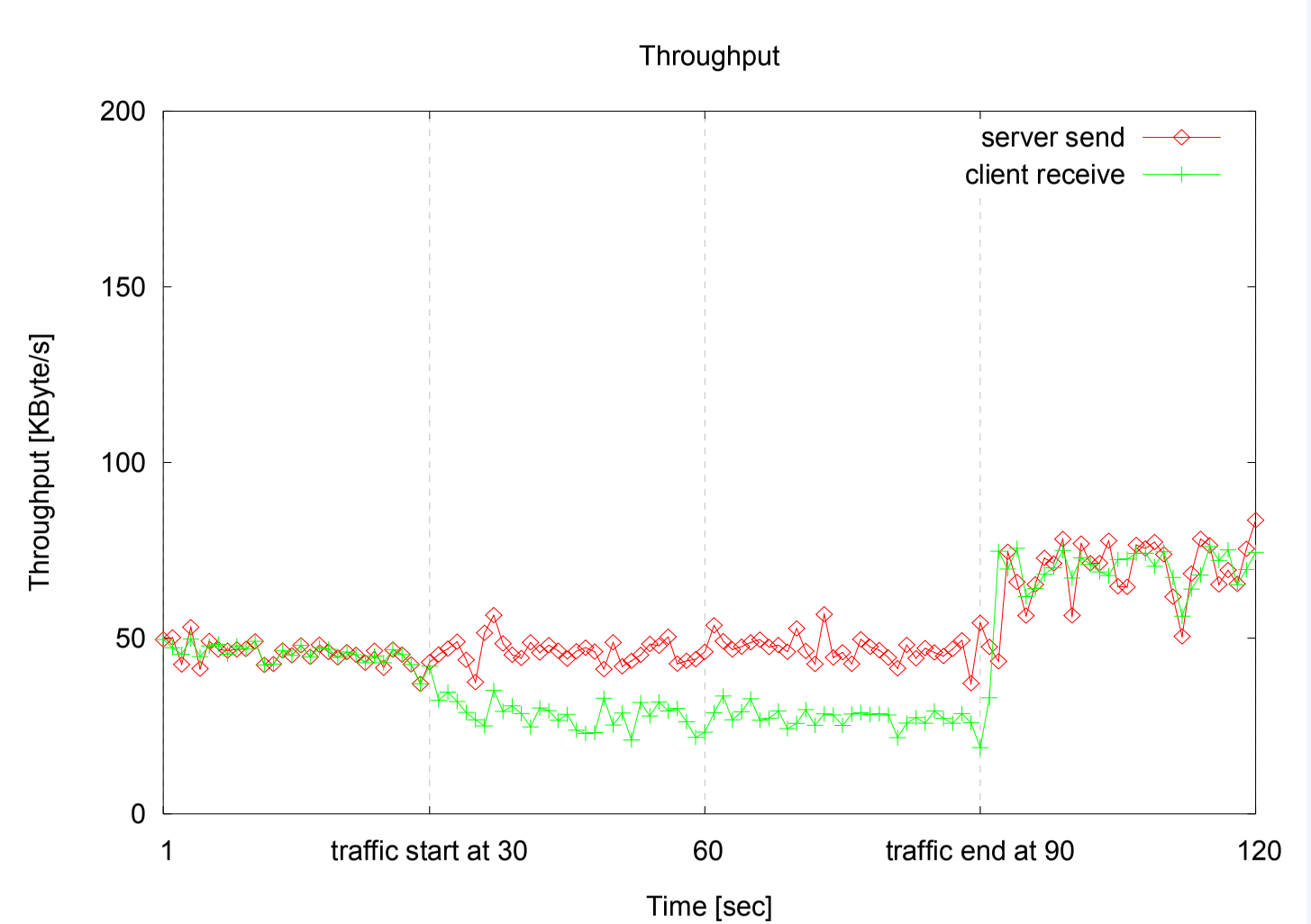
# TCP (the way it should be)



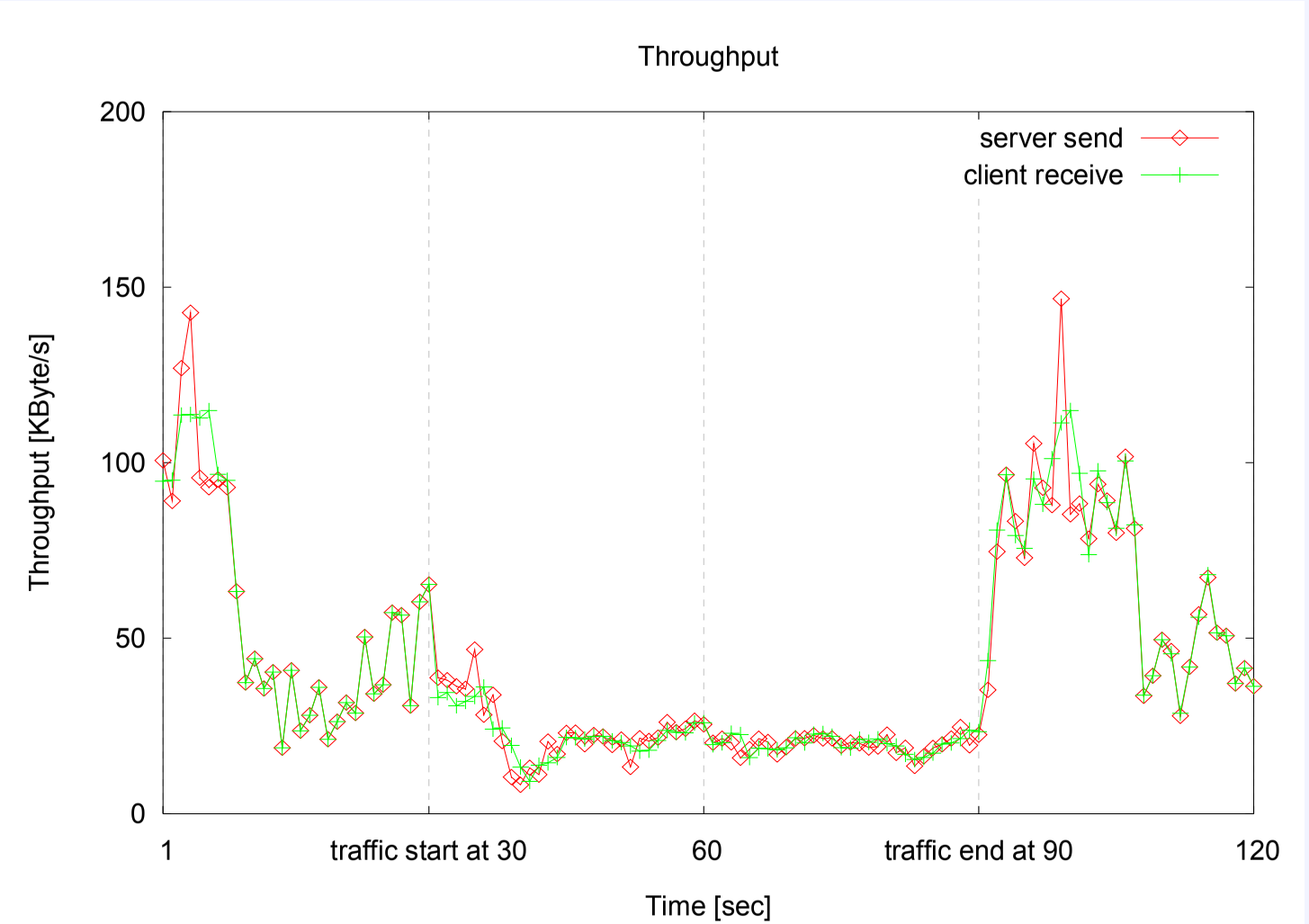
# Streaming Video: RealPlayer



# Streaming Video: Windows Media Player



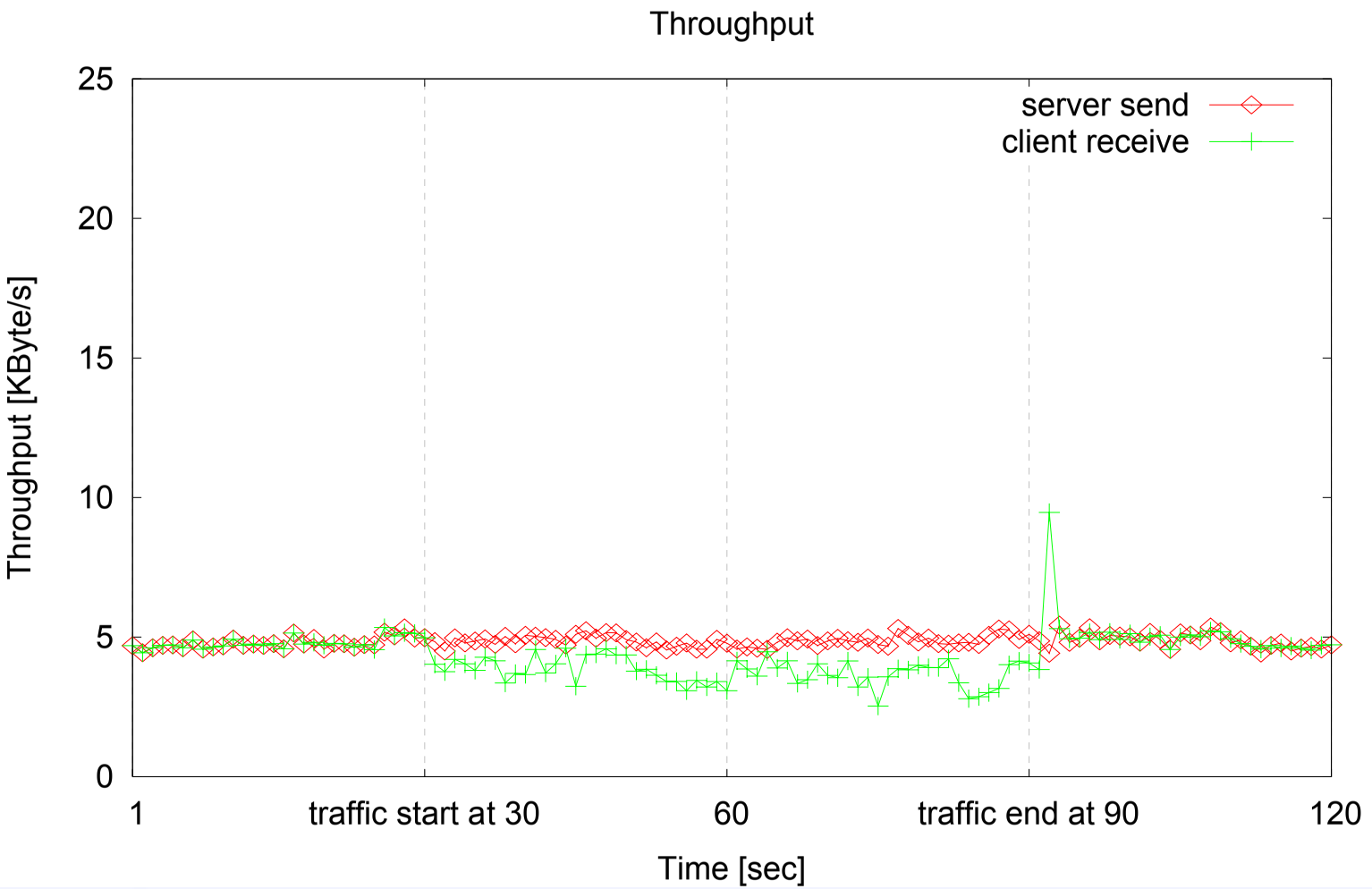
# Streaming Video: Quicktime



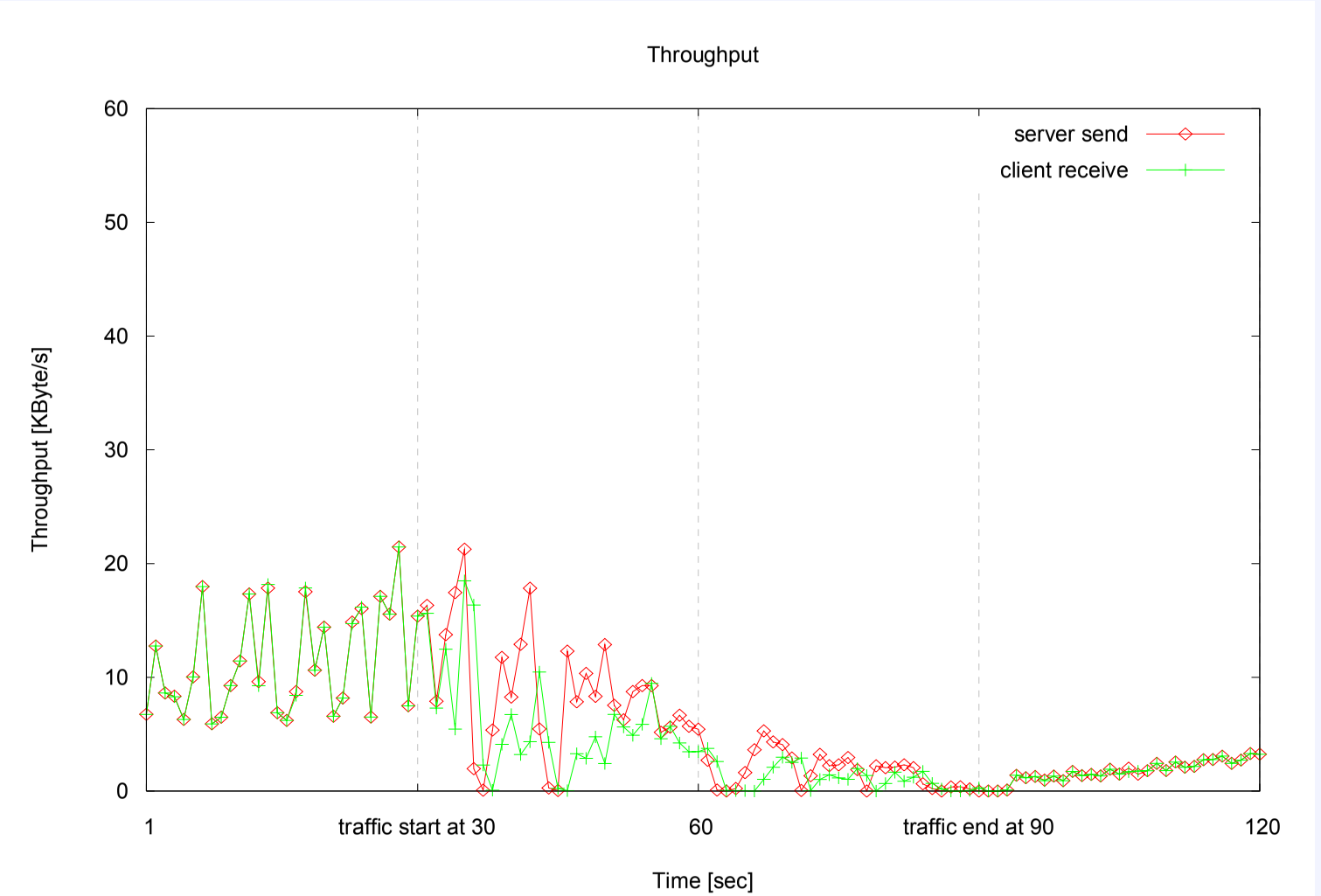
# VoIP: MSN


Throughput
server send
client receive
Throughput [KByte/s]
25
20
15
10
5
0
1
traffic start at 30
60
traffic end at 90
120
Time [sec]

# VoIP: Skype


Throughput
server send
client receive
Throughput [KByte/s]
25
20
15
10
5
0
1
traffic start at 30
60
traffic end at 90
120
Time [sec]

# Video conferencing: iVisit



# Observations

- Several other applications examined
  - ICQ, NetMeeting, AOL Instant Messenger, Roger Wilco, Jedi Knight II, Battlefield 1942, FIFA Football 2004, MotoGP2
- Often: congestion ⇒ increase rate
  - is this FEC?
  - often: rate increased by increasing packet size
  - note: packet size limits measurement granularity
- Many are unreactive
  - Some have quite a low rate, esp. VoIP and games
- Aggregate of unreactive low-rate flows = dangerous!
  - IAB Concerns Regarding Congestion Control for Voice Traffic in the Internet [RFC 3714]

# Some thoughts

- How TCP-friendly are 8 web browsers?
  - Congestion Manager: congestion control for all flows in OS core
  - MulTCP: Emulate multiple TCP's to provide differentiated services
- How TCP-friendly are short-lived flows? (web-traffic, ..)
- How to convince Internet multimedia app. programmers to implement TCP-friendly congestion control?
- Solution: Datagram Congestion Control Protocol (DCCP)
  - Well-defined framework for (TCP-friendly) congestion control
  - Sender app chooses an appropriate congestion control mechanism
  - Core OS implementation of mechanisms
  - Lots of additional features: nonces, partial / separate checksums (distinguish: corruption ⇔ congestion), ...

# What DCCP does for you (roughly)

- Multiplexing + protection against corruption
  - ports, checksum (UDP Lite ++)
- Connection setup and teardown
  - even though unreliable! one reason: middlebox traversal
- Feature negotiation mechanism
  - Features are variables such as CCID ("Congestion Control ID")
- Reliable ACKs ⇒ knowledge about congestion on ACK path
  - ACKs have sequence numbers
  - ACKs are transmitted (receiver) until ACKed by sender (ACKs of ACKs)
- Full duplex communication
  - Each sender/receiver pair is a half-connection; can even use different CCIDs!
- Some security mechanisms, several options

# Packet format

2 Variants; different sequence no. length, detection via X flag

| Source Port | | | | Destination Port |
|---|---|---|---|---|
| Data Offset | | CCVal | CsCov | Checksum |
| Res | Type | X = 1 | Reserved | Sequence Number (high bits) |
| Sequence Number (low bits) | | | | |

| Source Port | | | | Destination Port |
|---|---|---|---|---|
| Data Offset | | CCVal | CsCov | Checksum |
| Res | Type | X = 0 | Sequence Number (low bits) | |

- Generic header with 4-bit type field
  - indicates follwing subheader
  - only one subheader per packet, not several as with SCTP chunks

# Additional options

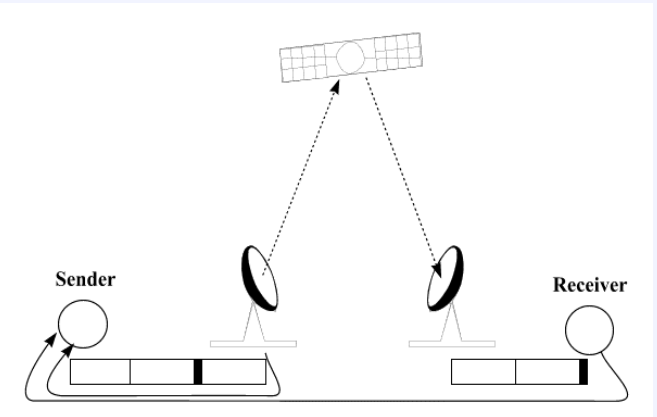
- Data Dropped: indicate differentdrop events in receiver (differentiate: not received by app / not received by stack)
  - removed from buffer because receiver is too slow
  - received but unusable because corrupt (Data Checksum option)
- Slow receiver: simple flow control
- ACK vector: SACK (runlength encoded)
- Init Cookie: protection against SYN floods
- Timestamp, Elapsed Time: RTT estimation aids
- Mandatory: next option must be supported
- Feature negotiation: Change L/R, Confirm L/R

# DCCP usage: incentive considerations

- Benefits from DCCP (perspective of a single application programmer)
  - ECN usage (not available in UDP API)
  - scalability in case of client-server based usage
  - TCP-based applications that are used at the same time may work better
  - perhaps smaller loss ratio while maintaining reasonable throughput
- Reasons not to use DCCP
  - programming effort, especially if it is an update to a working UDP based application
  - common deployment problems of new protocol with firewalls etc.
  - less total throughput than UDP
- What if dramatically better performance than UDP is required?
- Can be attained using "penalty boxes" - but:
  - requires such boxes to be widely used
  - will only happen if beneficial for ISP: financial loss from UDP unresponsive traffic > financial loss from customers whose UDP app doesn't work anymore
  - requires many apps to use DCCP
  - chicken-egg problem! Similar to QoS deployment towards end systems [RFC 2990]

# Some problems with TCP

- TCP over noisy links: problems with „packet loss = congestion“
  - was bad idea in times of error-prone networks
  - seems reasonable in times of fibre networks
  - really bad for wireless links!




- TCP over “long fat pipes“: large bandwidth*delay product
  - long time to reach equilibrium, MD = problematic!

- TCP in highly asymmetric networks: (e.g. direct satellite last mile)
  - incoming throughput (high capacity link) limited by rate of outgoing ACKs („ACK compression“)

# TCP++

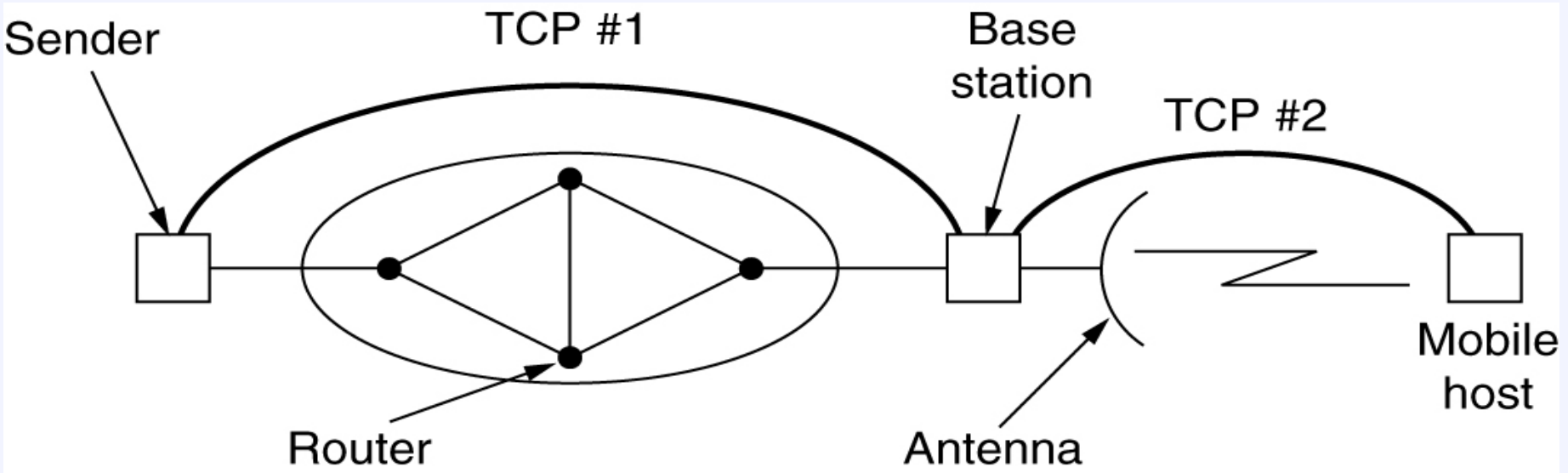
TCP over high speed links:

- larger initial window / window scaling option, TCP SACK
- Scalable TCP: increase/decrease functions changed
  (probing times proportional to rtt but not rate)
- HighSpeed TCP (merged with Scalable TCP):
  response function less drastic in high bandwidth environments *only*
- Quick-Start: query routers for initial sending rate with IP options
  Internet-draft

TCP over asymmetric links:

- ACK suppression, ACK compaction, TCP header compression

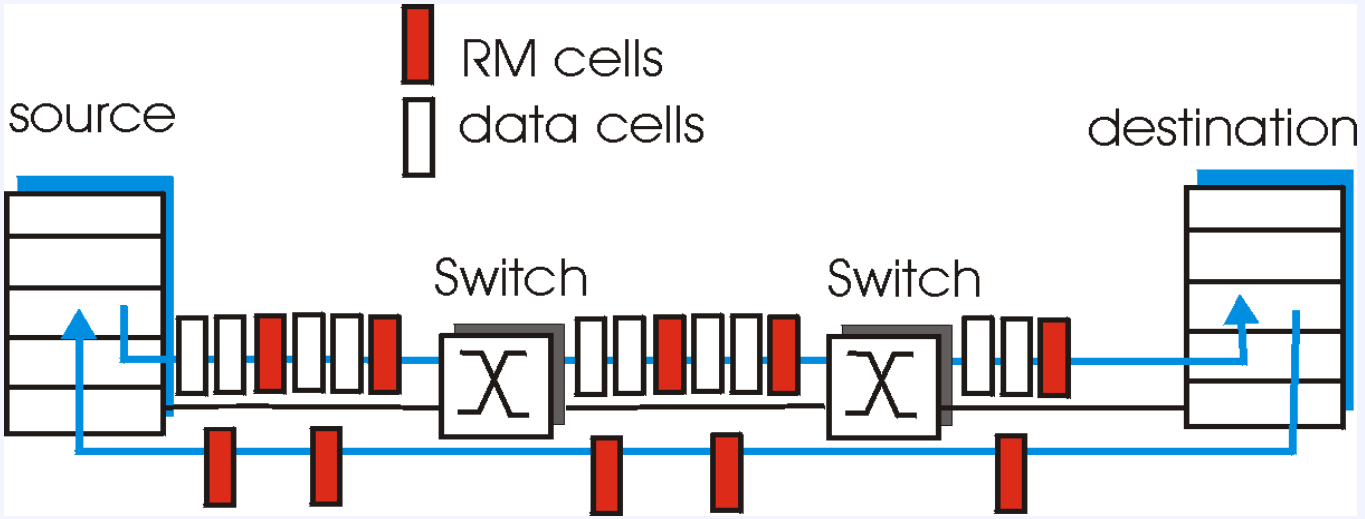
# TCP over noisy (wireless) links: PEP




- Various possible enhancements:
  - split connection at base station
  - monitor connection at base station, buffer + interfere ("Snoop TCP")
- Note: ECN is not affected by link noise!

# Beyond ECN

- ATM: Explicit Rate Feedback (part of Available Bit Rate (ABR) service)
  RM (resource management) cells:
  - sent by sender, interspersed with data cells; bits in RM cell set by switches
    - NI bit: no increase in rate (mild congestion), (EF)CI bit: like Internet ECN
    - two-byte ER (explicit rate) field: may be lowered by congested switch
    - sender' send rate thus minimum supportable rate on path!




- Problem: ATM failed (scalability? too much complexity in switches?)
- Experimental Internet approaches:
  - Multilevel ECN (two bits), eXpress Control Protocol (XCP), CADPC/PTP (my own)

# Other enhancements

- FAST TCP
  - Variant based on window and delay
  - Delay allows for earlier adaptation (awareness of growing queue)
  - Proven to be stable
  - Commercially announced + patent protected, by Steven Low's CalTech group
  - another delay-based example: TCP Vegas
- TCP Westwood+
  - different response function (proportional to rate)
  - proven to be stable
- Lots of experimental Active Queue Management schemes out there
  - Adaptive RED, BLUE, REM, RIO etc. etc.
- **Lots of open issues and problems waiting to be solved!**

# References

## General


- Raj Jain and K. K. Ramakrishnan, "Congestion Avoidance in Computer Networks with a Connectionless Network Layer: Concepts, Goals and Methodology", Proceedings of Computer Networking Symposium, Washington, D. C., April 11-13 1988, pp. 134-143.
- Van Jacobson, "Congestion Avoidance and Control", Proceedings of ACM SIGCOMM 1988, pp. 314-329.
- D. Chiu and R. Jain, "Analysis of the Increase/Decrease Algorithms for Congestion Avoidance in Computer Networks", Journal of Computer Networks and ISDN, Vol. 17, No. 1, June 1989, pp. 1-14.
- Sally Floyd and Van Jacobson, "On Traffic Phase Effects in Packet-Switched Gateways", Internetworking: Research and Experience, V.3 N.3, September 1992, p.115-156. Earlier version: Computer Communication Review, V.21 N.2, April 1991.
- Sally Floyd and Van Jacobson, "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, August 1993.
- K. Ramakrishnan, S. Floyd, and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

# References /2

Problems


- Scott Shenker, "Fundamental Design Issues for the Future Internet", IEEE Journal on Selected Areas in Communications, 13, pp. 1141-1149, 1995.
- Frank Kelly, "Charging and rate control for elastic traffic", European Transactions on Telecommunications, 8. pp. 33-37. An updated version is available at http://www.statslab.cam.ac.uk/frank/elastic.html
- Ramesh Johari, "Mathematical Modeling and Control of Internet Congestion", SIAM News, Vol. 33, No. 2.
- L. Massoulié and J. Roberts, "Bandwidth sharing: objectives and algorithms", Proceedings of IEEE Infocom 1999, New York City, New York, 21.-25. 3. 1999.
- Ramesh Johari and David Tan, "End-to-End Congestion Control for the Internet: Delays and Stability", IEEE/ACM Transactions on Networking 9 (2001) 818-832.
- Ashraf Matrawy and Ioannis Labadaris, "A Survey of Congesiton Control Schemes for Multicast Video Applications", IEEE Communications Survey & Tutorials, Fourth Quarter 2003, Vol. 5, No. 2

# References /3

## Proposed enhancements


- Mark E. Crovella and Azer Bestavros, "Self-similarity in World Wide Web Traffic: Evidence and Possible Causes", IEEE/ACM Transactions on Networking, Vol. 5, No. 6, December 1997.
- Andras Veres, Zsolt Kenesi, Sandor Molnar, Gabor Vattay, "On the Propagation of Long-range Dependency in the Internet", Proceedings of ACM SIGCOMM 2000, Stockholm, Sweden, August 28 - September 1 2000.
- Guanghui He, Yuan Gao, Jennifer C. Hou, Kihong Park, "A Case for Exploiting Self-Similarity of Network Traffic in TCP", 10th IEEE International Conference on Network Protocols (ICNP'02), Paris, France, Nov. 12-15, 2002.
- Jörg Widmer, Robert Denda, and Martin Mauve, "A Survey on TCP-Friendly Congestion Control", IEEE Network Magazine, Special Issue "Control of Best Effort Traffic" Vol. 15, No. 3, May 2001.
- Philippe Oechslin and Jon Crowcroft, "Differentiated End-to-End Internet Services using a Weighted Proportional Fair Sharing TCP", ACM Computer Communication Review (CCR), 1998.
- Hari Balakrishnan, Hariharan Rahul, and Srinivasan Seshan, "An Integrated Congestion Management Architecture for Internet Hosts", Proceedings of ACM SIGCOMM 1999, Cambridge, MA., September 1999.

# References /4

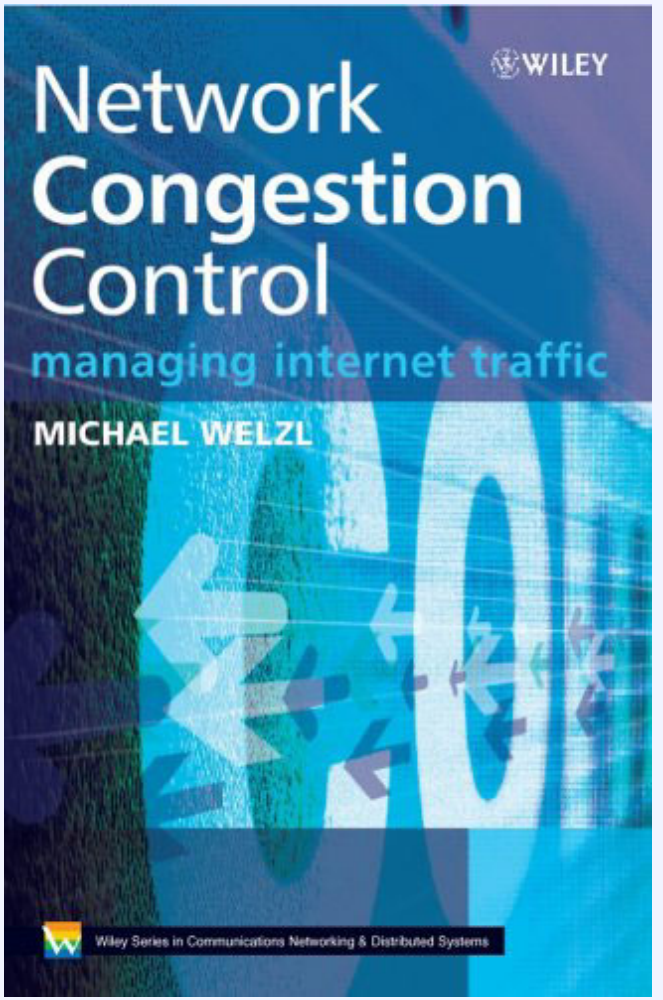

- H. Balakrishnan and S. Seshan, "The Congestion Manager", RFC 3124, June 2001.
- Eddie Kohler, Mark Handley, Sally Floyd, and Jitendra Padhye, "Datagram Congestion Control Protocol (DCCP)", Internet-draft (work in progress) draft-ietf-dccp-spec-05.txt, October 2003.
- Metz, C., "TCP Over Satellite... The Final Frontier.", IEEE Internet Computing, 1999.
- Sally Floyd, "HighSpeed TCP for Large Congestion Windows", RFC 3649, Experimental, December 2003.
- Balakrishnan, H., Padmanabhan, V. N., Seshan, S. and Katz, R. H., "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", Proceedings of ACM SIGCOMM 1996, Stanford, CA.
- Ramakrishnan, K., Floyd, S. and Black, D., "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168.
- Dina Katabi, Mark Handley, and Charlie Rohrs, "Congestion Control for High Bandwidth-Delay Product Networks", Proceedings of ACM SIGCOMM 2002, Pittsburgh, PA, 19-23 August 2002.
- Cheng Jin, David X. Wei and Steven H. Low, "FAST TCP: motivation, architecture, algorithms, performance", IEEE Infocom, March 2004.

# Main reference

Michael Welzl,

"Network Congestion Control:
Managing Internet Traffic",

John Wiley & Sons, Ltd., July 2005




Thank you!