

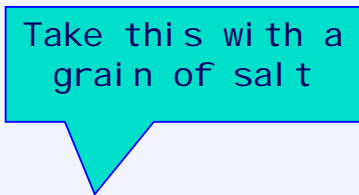
Internet congestion control: evolution and current open issues

Michael Welzl <http://www.welzl.at>
Institute of Computer Science
University of Innsbruck

CAIA guest talk
Swinburne Univ., Melbourne AUS
22 January, 2008

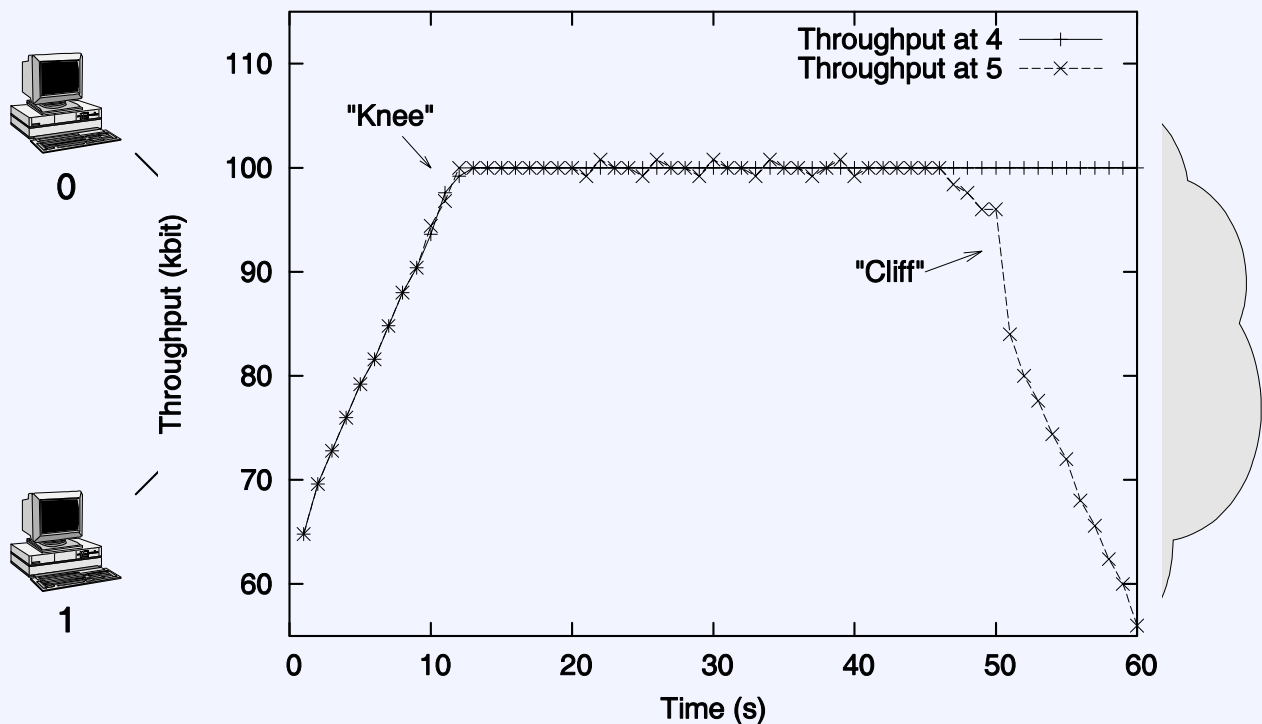
Outline

- Problem description (1986-1988)
 - congestion collapse
- Standard solution (1988-2004)
 - TCP congestion control in a nutshell
- Current problems & experimental improvements (2004-today)
- Reality check (today-future)
 - the role of the IRTF/IETF
 - Open issues



Take this with a
grain of salt

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.

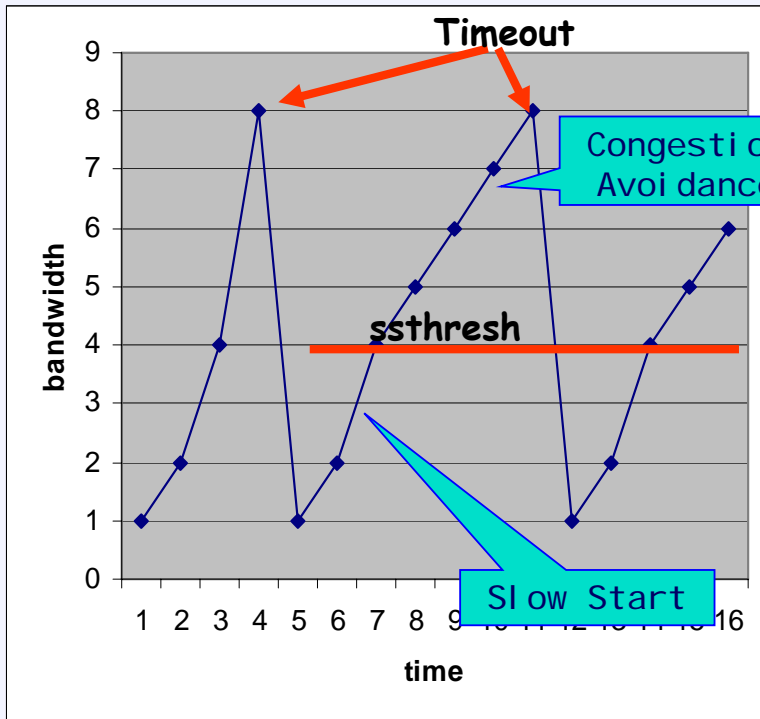
Internet congestion control: History

- 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 equilibrium, 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

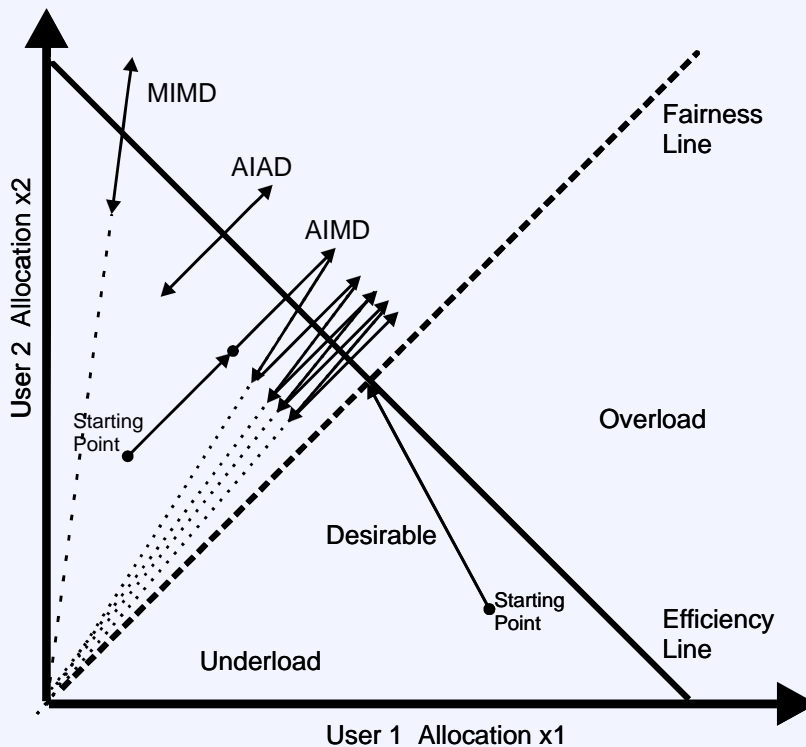
- Distinguish:
 - flow control: protect receiver against overload
(receiver "grants" a certain amount of data ("receiver window" (rwnd)))
 - congestion control: protect network against overload
("congestion window" (cwnd) limits the rate: $\min(cwnd, rwnd)$ used)
- Flow/Congestion Control combined in TCP. Two basic algorithms:
(window unit: SMSS = Sender Maximum Segment Size, usually adjusted to Path MTU;
init $cwnd \leq 2$ (*SMSS), ssthresh = usually 64k)
- Slow Start: for each ack received, increase cwnd by 1
(exponential growth) until $cwnd \geq ssthresh$
- Congestion Avoidance: each RTT, increase cwnd by at most one segment
(linear growth - "additive increase")
- Timeout: ssthresh = FlightSize/2 (exponential backoff - "multiplicative decrease"), $cwnd = 1$; FlightSize = bytes in flight (may be less than cwnd)

TCP Congestion Control /2

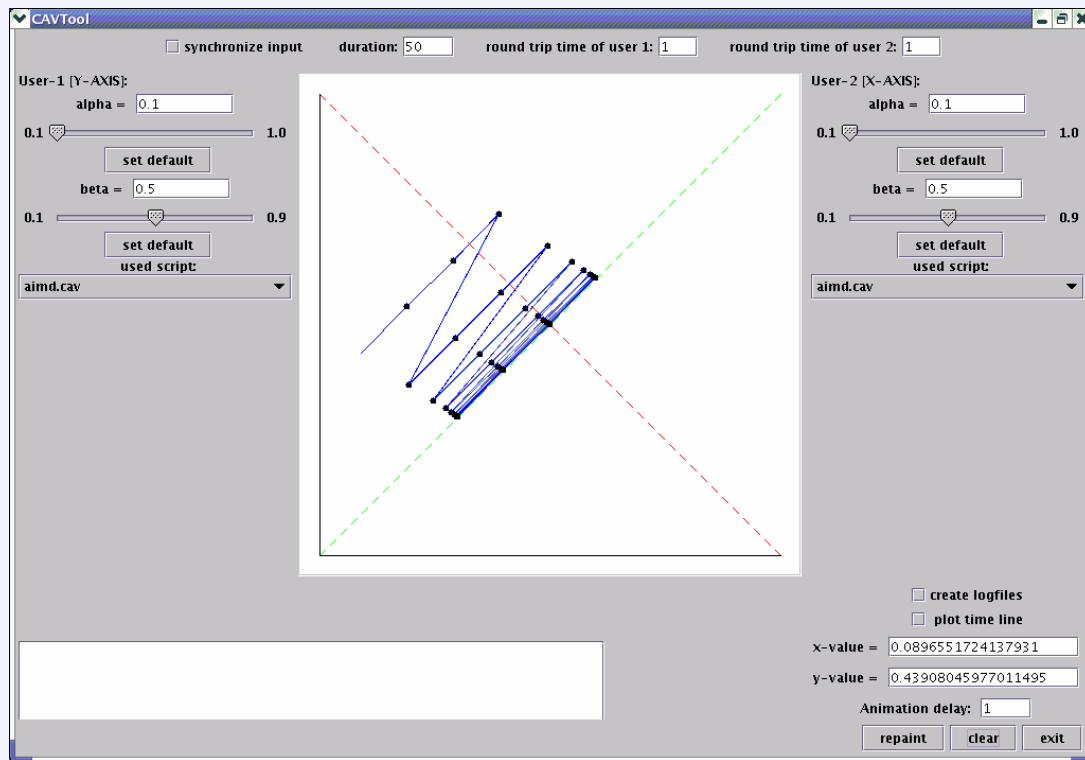


- If a packet or ack is lost (timeout), set $cwnd = 1$, $ssthresh = \text{current bandwidth} / 2$ ("multiplicative decrease") - exponential backoff
- Timeout based on RTT; good estimation is crucial!
- Later additions: *(TCP Reno, 1990)*
Fast retransmit / fast recovery (sender detects loss via duplicate acks)

Background: AIMD

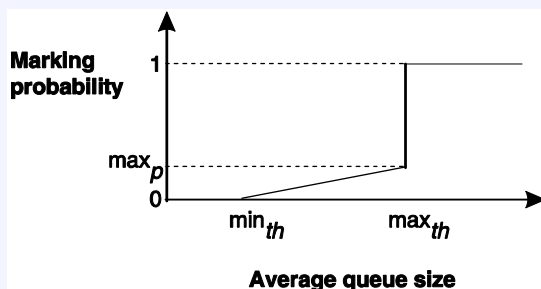


AIMD diagrams in action: Congestion Avoidance Visualization Tool (CAVT)

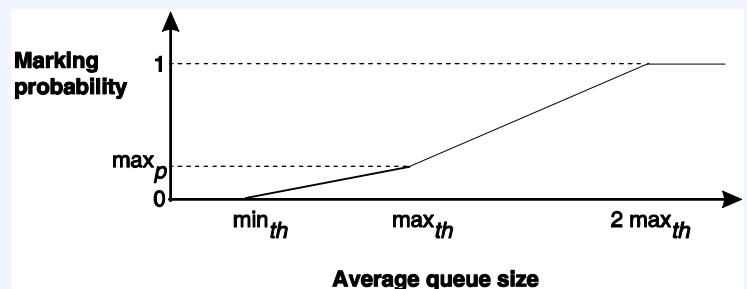


Active Queue Management

- Monitor queue, do not only drop upon overflow \Rightarrow more intelligent decisions
- Goals: eliminate phase effects, manage fairness ("punish" flows that are too aggressive)
 - Aggressive flows have more packets in the queue; thus, dropping a random one is more likely to affect such flows
 - Also possible to differentiate traffic via drop function(s)



RED



RED in "gentle" mode

Explicit Congestion Notification (ECN)

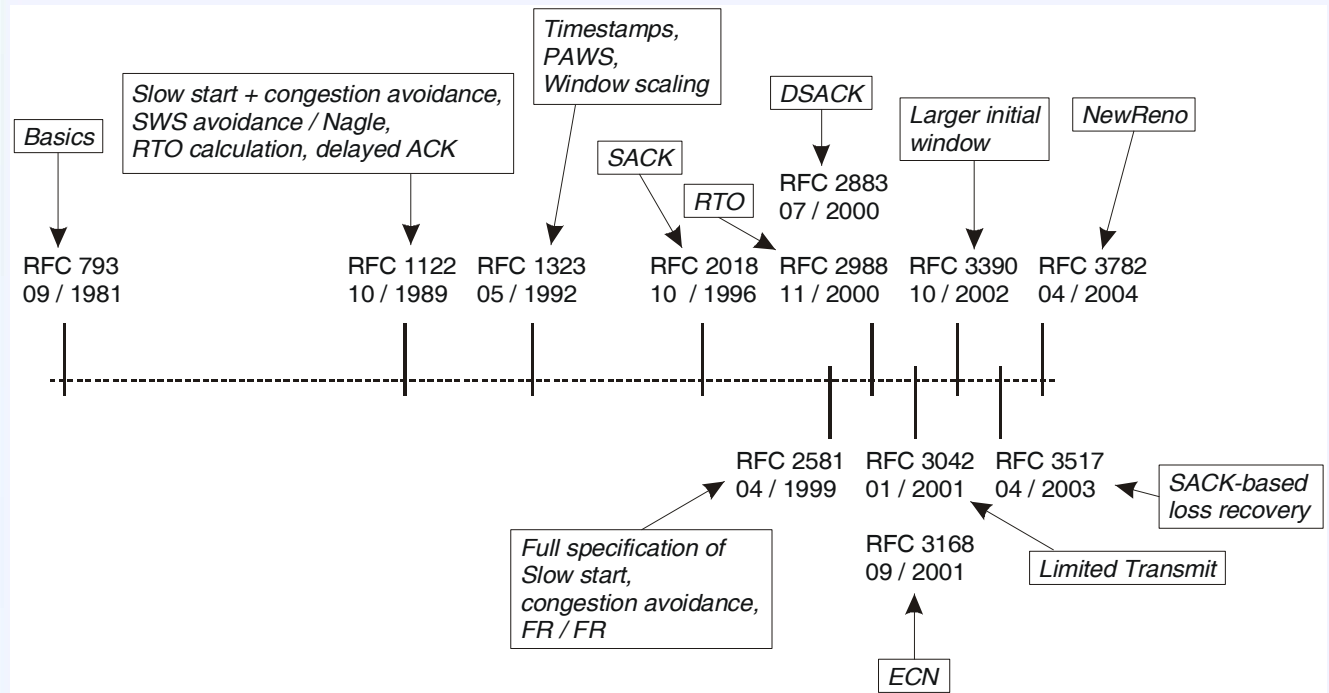
- Instead of dropping, set a bit
- Receiver informs sender about bit; sender behaves as if a packet was dropped
 - ⇒ actual communication between end nodes and the network
- Note: ECN = true congestion signal (i.e. clearly not corruption)
- ECN had deployment problems: broken firewalls
 - was disabled in Linux by default for a long time
- **ECN Nonce**: experimental proposal for preventing receiver from wrongly claiming that there was no congestion

Gradual IETF refinements of TCP

- Essentially corrections of wrong behavior; some examples below
- TCP should only reduce its rate once per RTT
 - but that may not work when multiple packets dropped from one window
 - as windows become larger, this becomes more important
 - **SACK**, **NewReno** and **Limited Transmit** alleviate this problem
- **Window Scaling Option**
 - rwnd field size limits sending rate in today's high speed environments
 - Solution: both sides agree to left-shift window value by N bit
- **Appropriate Byte Counting (ABC)**
 - by default, TCP increases once per ACK
 - Delayed ACK receiver: increase by at most 1 packet every 2 RTTs
 - Multiple small ACKs for one packet: increase faster
 - ABC fixes this

TCP over the years...

Standards track TCP RFCs which influence when a packet is sent (status: October 2007)



Current problems and experimental solutions

Control *what?* Traffic jams, huh?

- Nowadays, networks are often overprovisioned
 ⇒ 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"

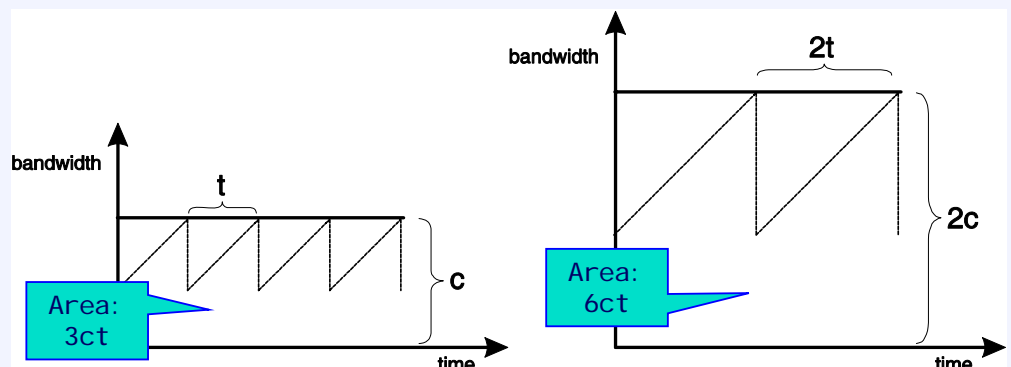
TCP with High Speed links

- TCP over "long fat pipes": large bandwidth*delay product
 - long time to reach equilibrium, MD = problematic!
 - From RFC 3649 (HighSpeed RFC, Experimental):

For example, for a Standard TCP connection with 1500-byte packets and a 100 ms round-trip time, achieving a steady-state throughput of 10 Gbps would require an average congestion window of 83,333 segments, and a packet drop rate of at most one congestion event every 5,000,000,000 packets (or equivalently, at most one congestion event every 1 2/3 hours). This is widely acknowledged as an unrealistic constraint.

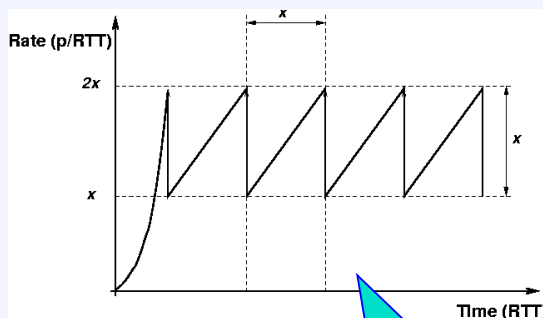
Theoretically,
utilization
independent of
capacity

But: longer
convergence time

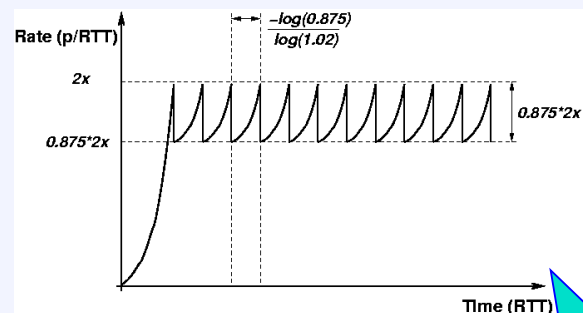


Proposed solutions

- Standards: larger initial window / window scaling option, TCP SACK
- Scalable TCP: increase/decrease functions changed
 - $\text{cwnd} := \text{cwnd} + 0.01$ for each ack received while not in loss recovery
 - $\text{cwnd} := 0.875 * \text{cwnd}$ on each loss event (probing times proportional to rtt but not rate)



Standard TCP



Scalable TCP

Source: <http://www.deneholme.net/tom/scalable/>

Proposed solutions /2

Rate	Standard TCP recovery time	Scalable TCP recovery time
1Mbps	1.7s	2.7s
10Mbps	17s	2.7s
100Mbps	2mins	2.7s
1Gbps	28mins	2.7s
10Gbps	4hrs 43mins	2.7s

- HighSpeed TCP (RFC 3649 includes Scalable TCP discussion):
 - response function includes $a(\text{cwnd})$ and $b(\text{cwnd})$, which also depend on loss ratio
 - less drastic in high bandwidth environments with little loss *only*
 - **Significant step!**
 - Previously, either TCP-friendly or better-than-TCP; no combinations!
- TCP Westwood+
 - different congestion response function (proportional to rate instead of $\beta = 1/2$)
 - Proven to be stable, tested in real life experiments, available in your Linux

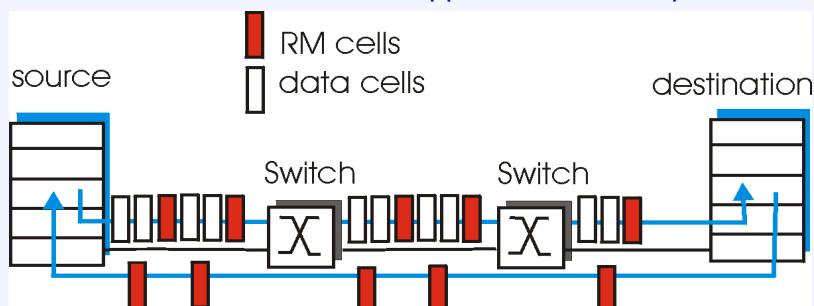
Proposed solutions /3

- FAST TCP, HTCP
 - Variants based on window and delay
 - Delay allows for earlier adaptation (awareness of growing queue)
 - Proven to be stable
 - FAST was commercially announced + patent protected, by Steven Low's CalTech group
 - based on an older delay-based TCP variant: TCP Vegas
 - Vegas = impractical because less aggressive than standard TCP

- BIC, CUBIC
 - BIC (Binary InCrease TCP) uses binary search to find the ideal window size:
 - when loss occurs, current window = max, new window = min
 - check midpoint;
 - if no loss \Rightarrow new min, increase; else new window = new max
 - CUBIC = BIC++ using cubic function; growth does not depend on RTT

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!



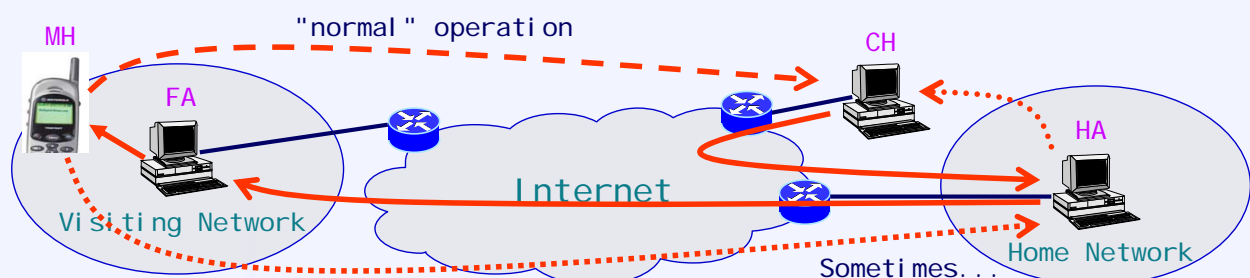
- Experimental Internet approaches:
 - Multilevel ECN (two bits), eXpress Control Protocol (XCP), CADPC/PTP (my own)
 - Quick-Start: query routers for initial sending rate with IP options
 - IETF effort; many discussions: security (nonces again), IP option handling
 - Routers often drop or delay packets with options; thus, suggested for controlled environments only

TCP in noisy environments

- TCP over noisy links: problems with "packet loss = congestion"
 - Usually wireless links, where delay fluctuations from link layer ARQ and handover are also issues (mitigation: spurious timeout detection schemes)
- TCP HACK, TCP Corruption Notification Options
 - Similar to DCCP Data Checksum Option: additional checksum over payload
 - Enables differentiating corruption / congestion
 - Correct reaction unknown (Lachlan Andrew has a proposal)
- Explicit Transport Error Notification (ETEN)
 - Use signaling protocol to query for noise ratio
 - Update rate based on this additional feedback
- Loss Tolerant TCP (LT-TCP)
 - K. K. Ramakrishnan et al: hybrid ARQ/FEC scheme
 - only ECN signals interpreted as definite congestion indications

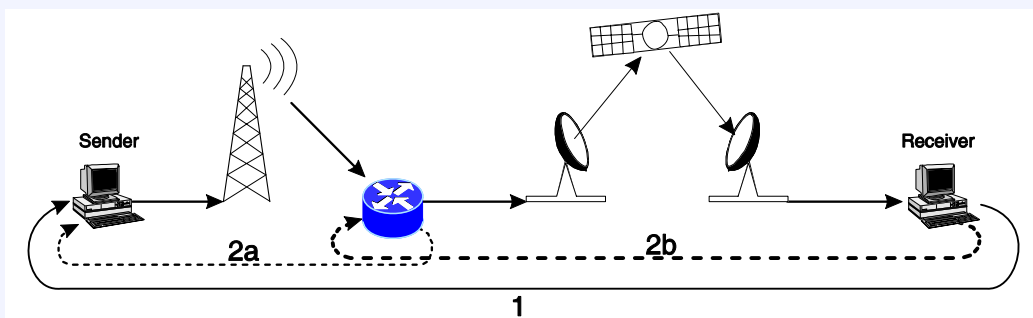
TCP with asymmetric routing

- TCP in asymmetric networks
 - incoming throughput (high capacity link) can be limited by rate of outgoing ACKs (ACK compaction, ACK congestion)
 - Mitigation:
 - Delayed ACKs
 - ACK suppression (selectively drop ACKs)
 - TCP header compression
 - triangular routing with Mobile IP(v4) and FA-Care-of-address can lead to unnecessarily large RTT (and hence large RTT fluctuations)



TCP over Satellite and PEPs

- Satellites combine several problems
 - Long delay
 - High capacity
 - Wireless (but usually not noisy (for TCP) because of link layer FEC)
 - Can be asymmetric (e.g. direct satellite downlink, 56k modem uplink)
- Thus, TCP over satellite is a major research topic
 - Transparent improvements ("Performance Enhancing Proxies") common
 - Figure: split connection approach: 2a / 2b instead of control loop 1
 - Many possibilities - e.g. Snoop TCP: monitor + buffer; in case of loss, suppress DupACKs and retransmit from local buffer



Active Queue Management gallery

Mechanism	What is monitored?	What is done?
RED, Adaptive RED, DRED	queue length	Packets are randomly dropped based upon the average queue length
SRED	queue length, packet header	Flow identifiers are compared with random packets in a queue history ('zombie list'); this is used to estimate the number of flows, which is an input for the drop function
BLUE	packet loss, 'link idle' events	The drop probability is increased upon packet loss and decreased when the link is idle
SFB	packet loss, 'link idle' events, packet header	Packets are hashed into bins, BLUE is applied per bin, the minimum loss probabilities of all bins that a packet is hashed into is taken; it is assumed to be very high for unresponsive flows only

very rough overview

Mechanism	What is monitored?	What is done?
AVQ	packet arrival rate	A virtual queue is maintained; its capacity is updated based on packet arrival rate
RED-PD	queue length, packet header, packet drops	The history of packet drops is checked for flows with a high rate; such flows are monitored and specially controlled
FRED	queue length, packet header	Flows that have packets buffered are monitored and controlled using per-flow thresholds
CHOKe	queue, packet header	If a packet is from the same flow as a randomly picked one in the queue, both are dropped
REM	arrival rate (optional), queue length	A 'price' variable is calculated based on rate (too low?) and queue (too high?) mismatch; the drop probability is exponential in price, the end-to-end drop probability is exponential in the sum of all link prices

Reality check and the role of the IRTF/IETF

Deployment of high speed TCPs

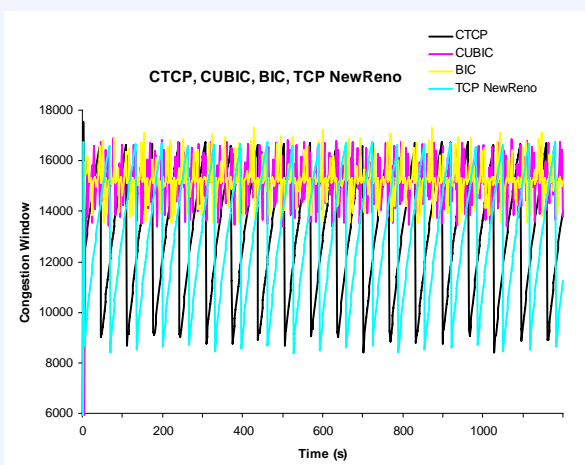
- High-speed TCP proposals have been on the table for quite a while
 - IETF did nothing: conservative about changing TCP
 - So people started using experimental mechanisms themselves
- Many mechanisms have long been available in Linux (pluggable CC)
 - pluggable CC soon also available in FreeBSD
- After major press release (Slashdot: "BIC-TCP 6000 times quicker than DSL"), **BIC became default TCP CC. in Linux in mid-2004**
 - Now replaced with CUBIC
- Compound-TCP (CTCP) = default TCP CC. in Windows Vista Beta
 - For testing purposes; disabled by default in standard release
- Will this lead to an arms race?

The role of the IRTF / IETF

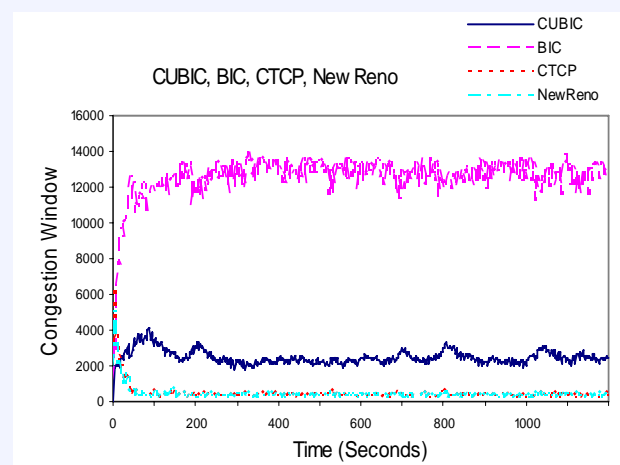
- The IETF wants interoperable mechanisms, specified in RFCs
 - so, authors of TCP proposals should be asked to specify their mechanisms
- Process devised: proposals will be pre-evaluated by IRTF Internet Congestion Control Research Group (ICCRG)
 - Evaluation guidelines: RFC 5033, Transport Models Research Group (TMRG)
 - CTCP and CUBIC proposals currently on the table (October 2007)
 - See: <http://www.irtf.org/charter?gtype=rg&group=iccr> for more details
- Procedure
 1. Write a draft
 2. Get reviews in the IRTF ICCRG; reviewers should check:
 - Does the proposal have a conflict with draft-floyd-tsvwg-cc-alt?
 - Were the TMRG metrics used in performance evaluations?
 3. Then go to the IETF, where reviews should be taken into account
- But that doesn't really solve all problems...

("Flow Rate") Fairness

- Common approach for making a mechanism work in the Internet: become less aggressive (with standard TCP at the far end of the spectrum) as loss increases
 ⇒ congestion collapse won't happen.
- But, in the little loss regime...

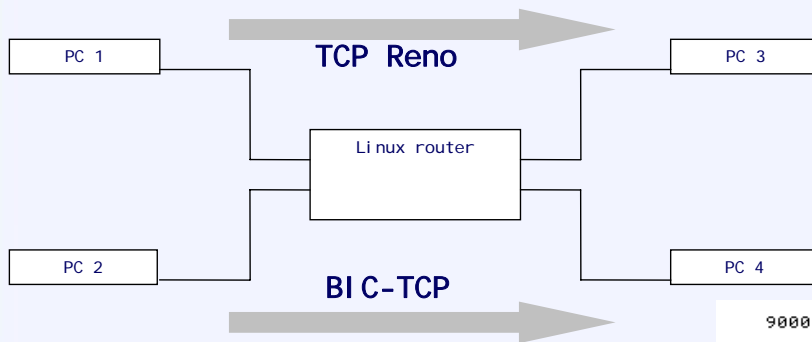


One mechanism at a time



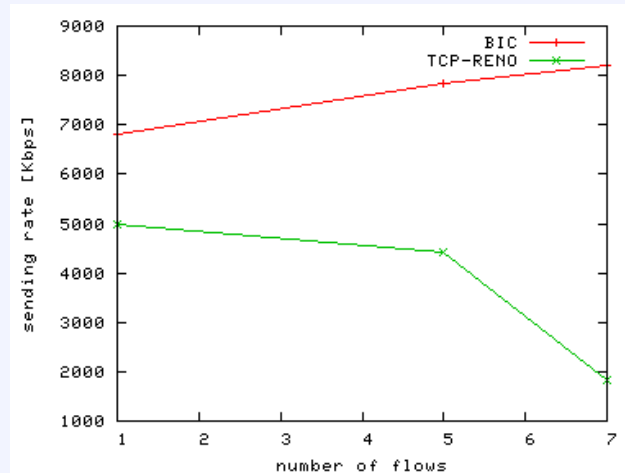
Mixed Mechanisms

Measurements in a local testbed



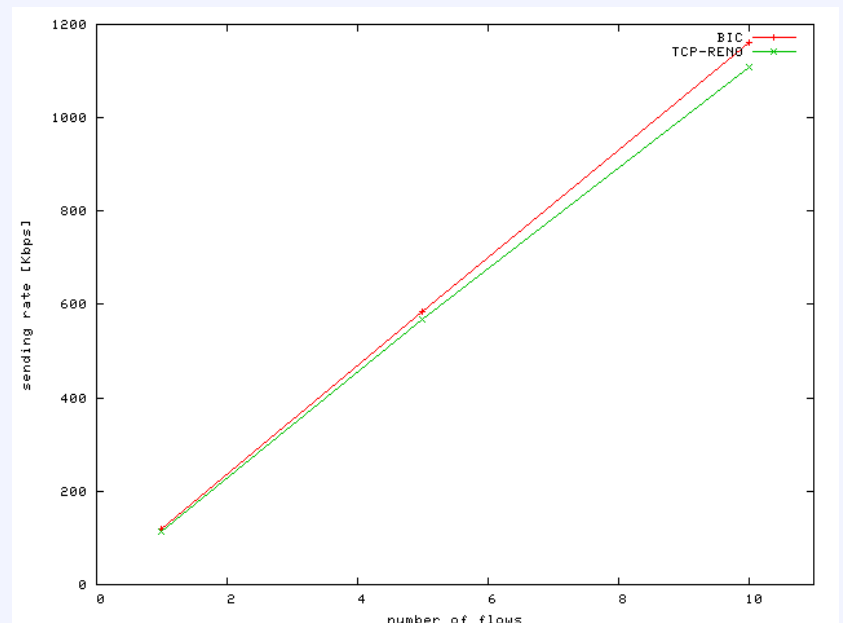
Fast Ethernet (100 Mbit/s)
All PCs running RedHat 8.0,
Kernel v2.4.18

- Doesn't look good
- but CUBIC supposedly less aggressive



But in fact, there is a bigger problem...

- PlanetLab measurements look quite different from local ones
- Why is that?
- Window Scaling not supported \Rightarrow rwnd limits sending rate
- 10 TCPs get exactly 10 times as much as 1
- So who cares about congestion control?



Depressing, isn't it?

- I raised this point in the IRTF e2e-interest mailing list - excerpts from answers:
- Glen Turner:
 - The problem is well described at <http://lwn.net/Articles/92727/> and in the threads at <http://oss.sgi.com/archives/netdev/2004-07/msg00146.html> , <http://kerneltrap.org/node/6723>
 - The known faulty equipment is:
 - Cisco PIX NAT feature corrupting in presence of SACK and window scaling. I don't have a Cisco bug ID for that - the Cisco bug navigator requires the specific version of software to be known to hunt for a bug, which makes finding historical bugs hard. You would presume that people kept their firewall software up-to-date, but the PIX had a bug where it filtered packets with IP.ECN != 00 and that took years to disappear.
 - Linux routers running the Netfilter firewalling package with the tcp-window-tracking module from the Netfilter Patch-o-matic. This bug was fixed in May 2003 <http://oss.sgi.com/archives/netdev/2004-07/msg00261.html> but made it into a lot of domestic appliance firewall/routers in 2002-4. Workaround is to disable firewall, fix is to upgrade software (which may not be possible since many manufacturers don't support older models and the source code for self-support is often not available, despite the GPL).
 - It is suspected that other faults exist, simply because of the number of bandwidth-shaping middleboxes which munge with the TCP window.

E2E-RG window scaling answers, cont'd

- Lars Eggert:
 - Microsoft presented their findings related to window scaling (and several other TCP extensions) at the IETF TSVAREA meeting in Prague. See <http://www3.ietf.org/proceedings/07mar/slides/tsvarea-3/sld3.htm> and the two following slides.
 - Summary: Window scaling is enabled in Vista, but limited to a factor of 2.
- David Reed:
 - It's fascinating to me that Window Scaling (an end-to-end option) would be screwed by bugs in *routers*. If literally true about network layer routers, what that means is that the whole design of the Internet is now beyond modification, since the modularity that modification depends on cannot be presumed.
 - So I'm even more depressed than Michael.

Other open issues (from an ICCRG meeting)

- Reaction to corruption (*DCCP spec asking*)
 - Note: corruption and congestion can be heavily correlated on short time-scales, and links can have strange properties (e.g. HSDPA, 802.11B)
- TCP over IETF mobility / ad hoc protocols (example: *draft-schuetz-tcpm-tcp-rlci*)
 - Can we show that the problem space is equal to another one, e.g. load changing on a single path?
- Evaluation of (implicit and explicit) feedback signals
 - Interactions with QoS, Traffic Engineering (real-time), IPsec, lower layers, congestion = f(bytes or packets?)
- Pseudowires
 - E.g., some consume bandwidth independent of the payload (*Pseudowire WG charter mentions CC, but drafts and RFCs restrict use to dedicated paths because proper CC unknown*)

Other open issues (from an ICCRG meeting) /2

- WG on pre-congestion notification
- Precedence for elastic traffic (related to MLPP docs, there may be a BOF soon)
- Misbehavior of senders and receivers (*TCPM discussions*), Denial-of-Service
- What is effective for media streams (*RTP profiles*)
- UDP based application layer protocols (*IRIS, SYSLOG - Sally Floyd's congestion control recommendation RFC is too unspecific for these groups*)
- Congestion control at the application layer (*SIP overload, ETSI GOCAP*)

Conclusion

- Congestion control problem has changed
 - from: there is congestion, what do we do?
 - via: networks are empty, what do we do?
 - to: how do we get all this stuff deployed and let it interoperate?
- Plenty of other open issues in congestion control
 - Corruption, multimedia streams, ideal type of feedback, ...
- After 20+ years, this is still an interesting topic, and quite important for the Internet
- IRTF ICCRG is not only a reviewing body; charter is quite broad
 - interesting proposals are more than welcome!

Thank you!

Questions?