Gipfelkonferenz 2006

Schnellere Netze durch MEP-VPN

phion Forschungskooperation mit Uni Innsbruck

Michael Welzl, http://www.welzl.at

Outline

- What's wrong?
  - TCP congestion control doesn't work very well
  - TCP has too many features
  - ...but UDP doesn't do enough
- How will we fix it?
  - VPN scenario with phion devices, enabling us to use...
  - SCTP
  - New congestion control mechanisms
  - Control of UDP

Transport today: one size fits all

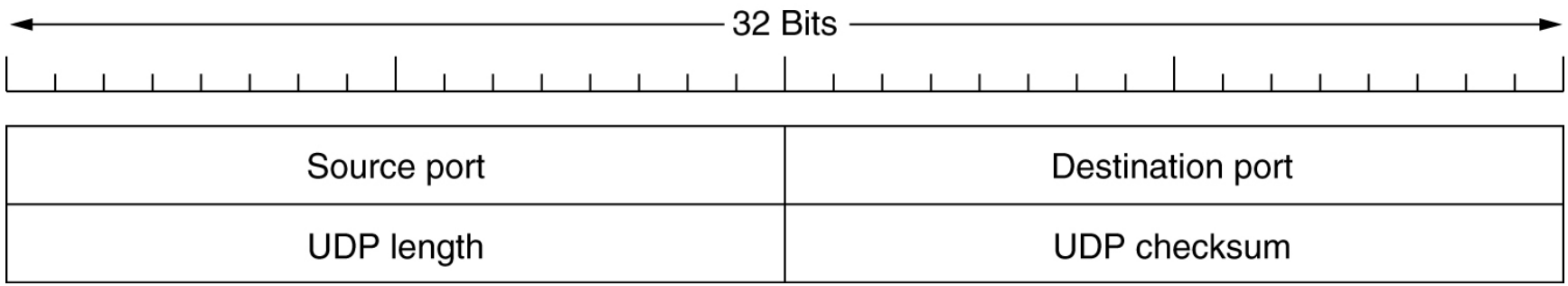
- UDP used for sporadic messages (DNS) and some special apps
- TCP used for everything else
  - now approximately 83 % according to:
    Marina Fomenkov, Ken Keys, David Moore and k claffy, "Longitudinal study of Internet traffic in 1998-2003", CAIDA technical report, available from http://www.caida.org/outreach/papers/2003/nlanr/
  - backbone measurement from 2000 said 98% ⇒ UDP usage growing
- Original Internet proposition:
  IP over everything, everything over IP
- Today's reality:
  IP over everything, almost everything over TCP, and the rest over UDP

What TCP does for you (roughly)

- UDP features: multiplexing + protection against corruption
  - ports, checksum
- stream-based in-order delivery
  - segments are ordered according to sequence numbers
  - only consecutive bytes are delivered
- reliability
  - missing segments are detected (ACK is missing) and retransmitted
- flow control
  - receiver is protected against overload (window based)
- connection handling
  - explicit establishment + teardown
- full-duplex communication
  - e.g., an ACK can be a data segment at the same time (piggybacking)
- congestion control
  - network is protected against overload (window based)
  - protocol tries to fill available capacity

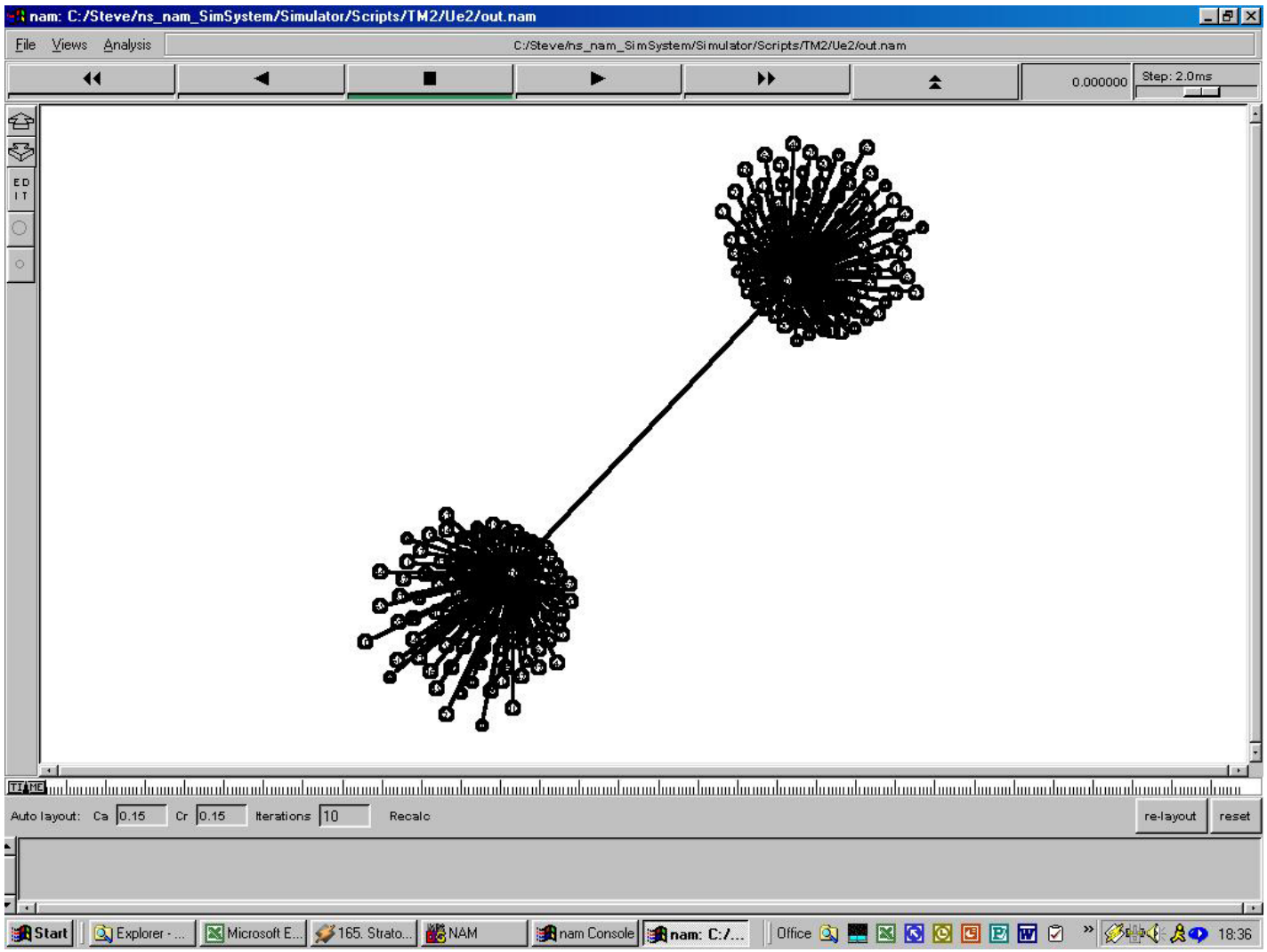
Are all these features always appropriate?

UDP, however...

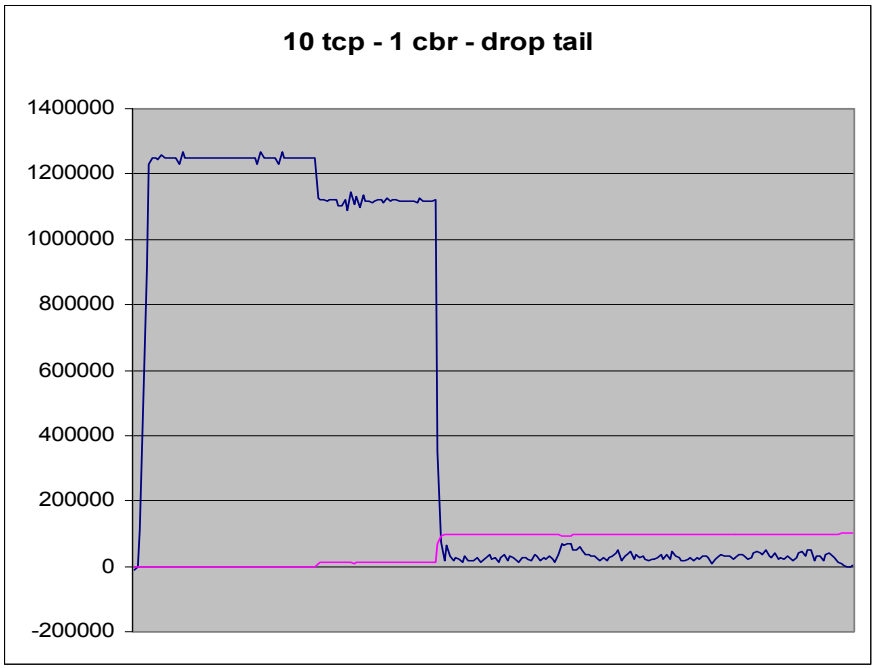


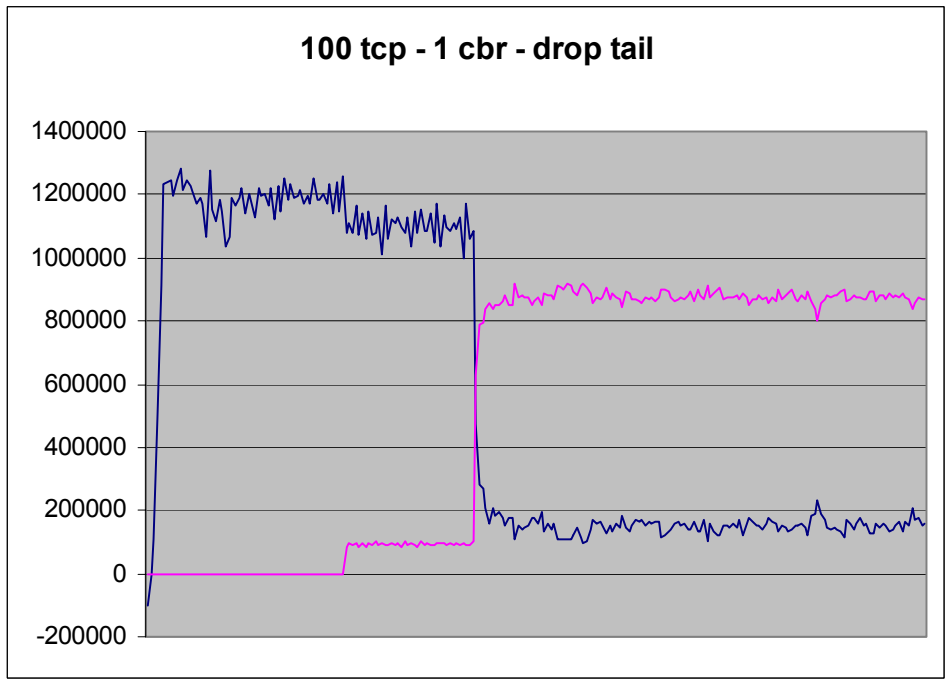
- RFC 768: three pages!
- IP + 2 features:
  - Multiplexing (ports)
  - Checksum
- Used by apps which want unreliable, timely delivery
  - e.g. VoIP: significant delay = ☹ ... but some noise = ☺
- No congestion control
  - fine for SNMP, DNS, ..

TCP vs. UDP: a simple simulation example



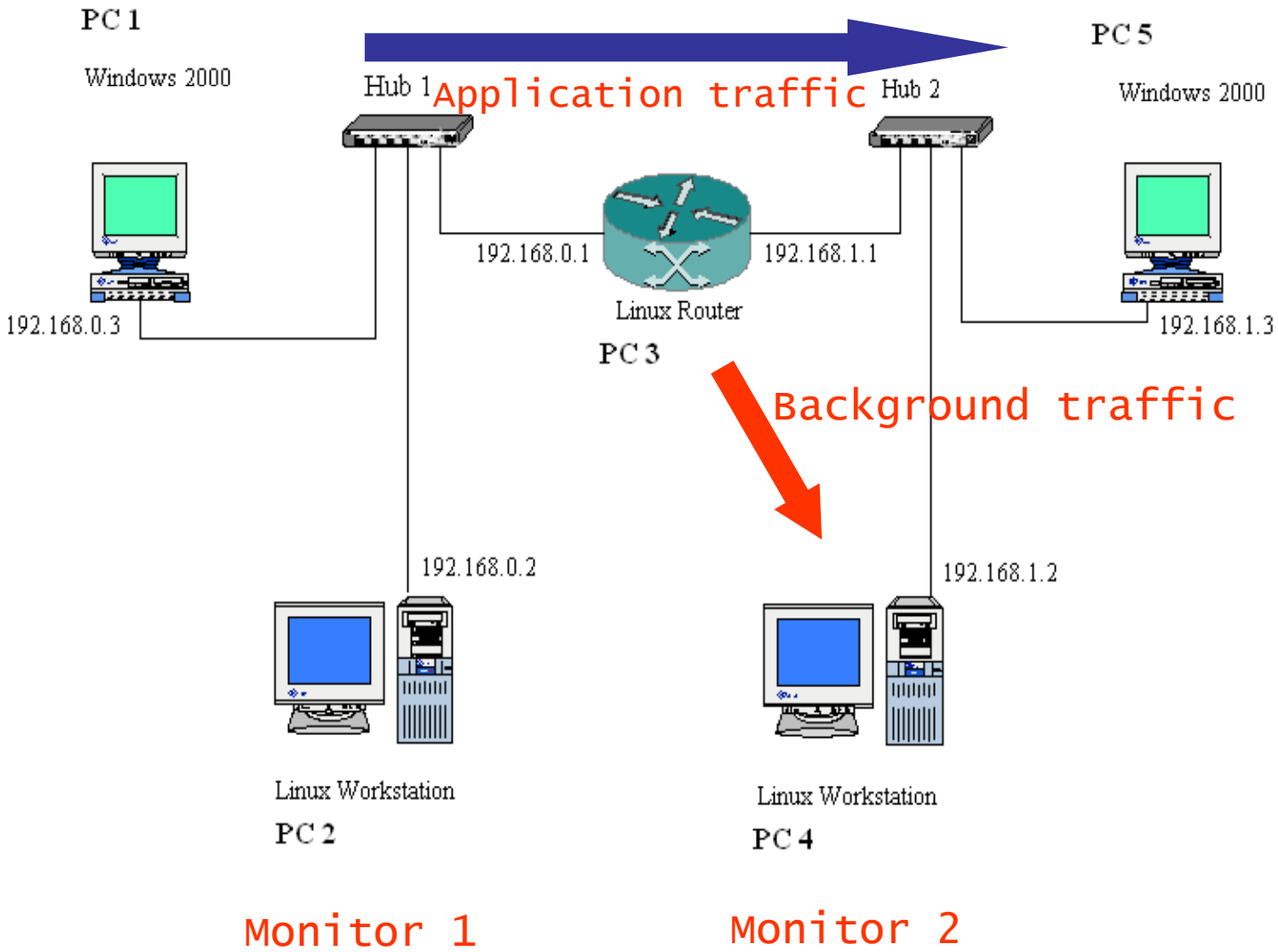
It doesn‘t look good



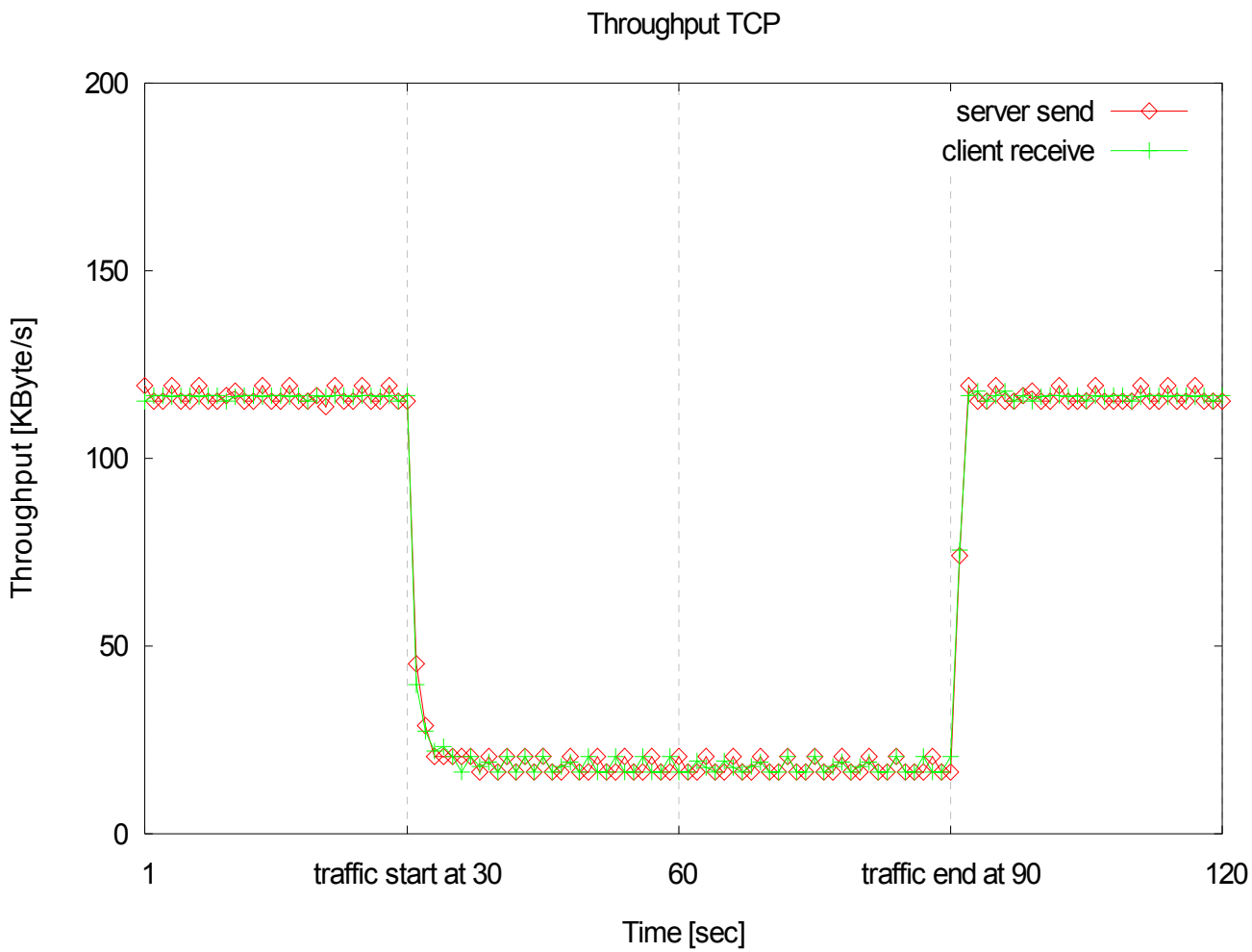


- For more details, see:
  Promoting the Use of End-to-End Congestion Control in the Internet.
  Floyd, S., and Fall, K..
  IEEE/ACM Transactions on Networking, August 1999.

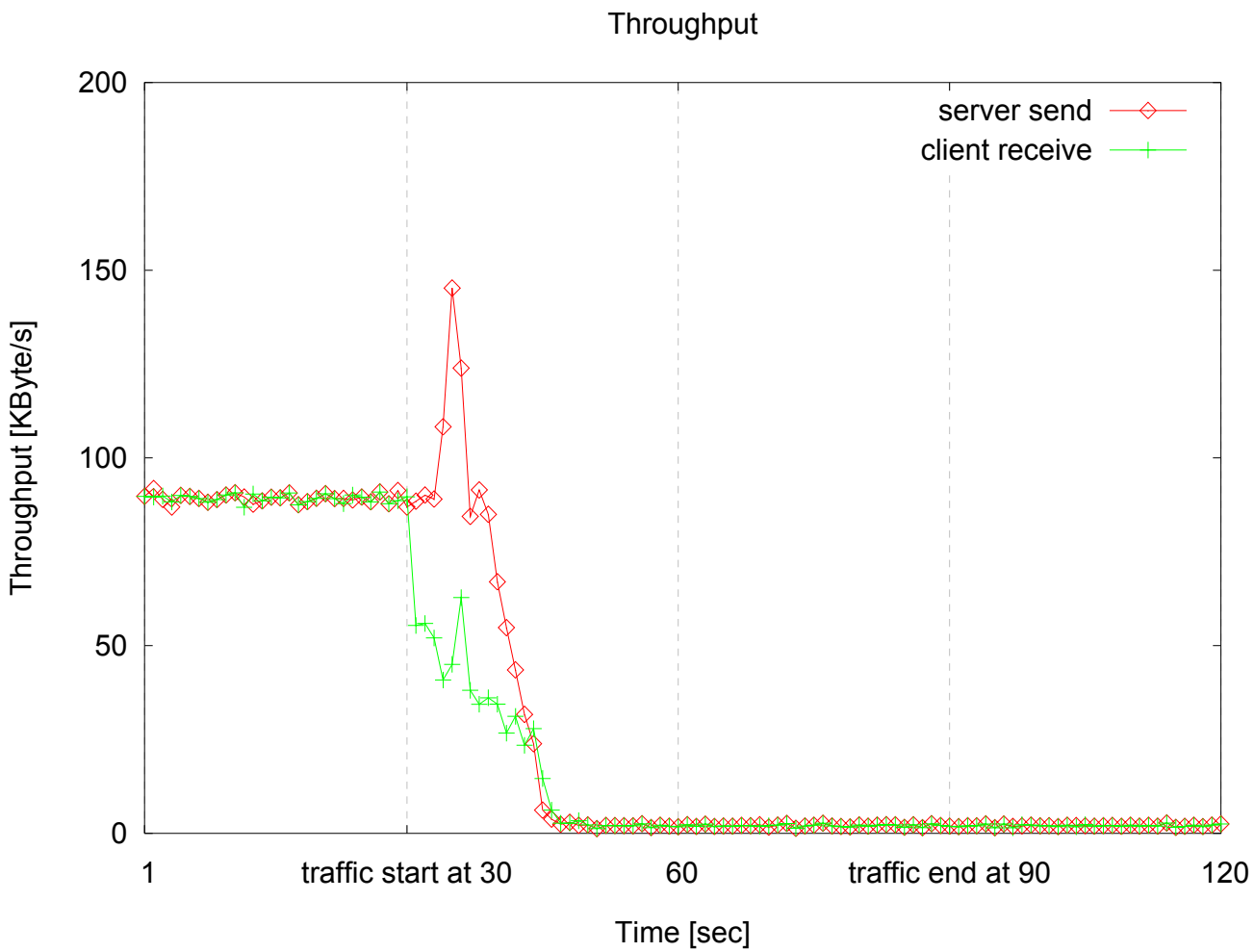
Real behavior of today‘s apps



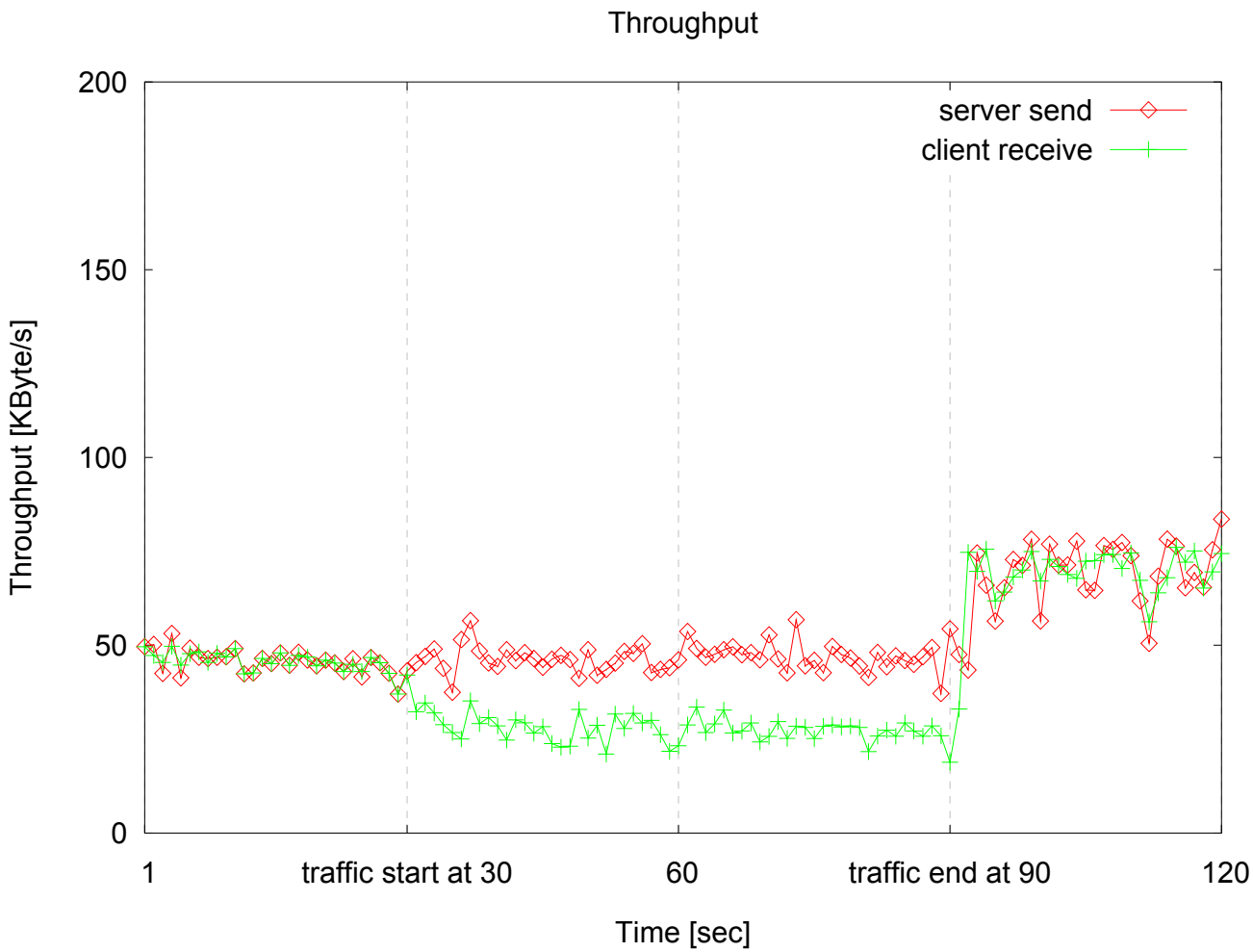
TCP (the way it should be)



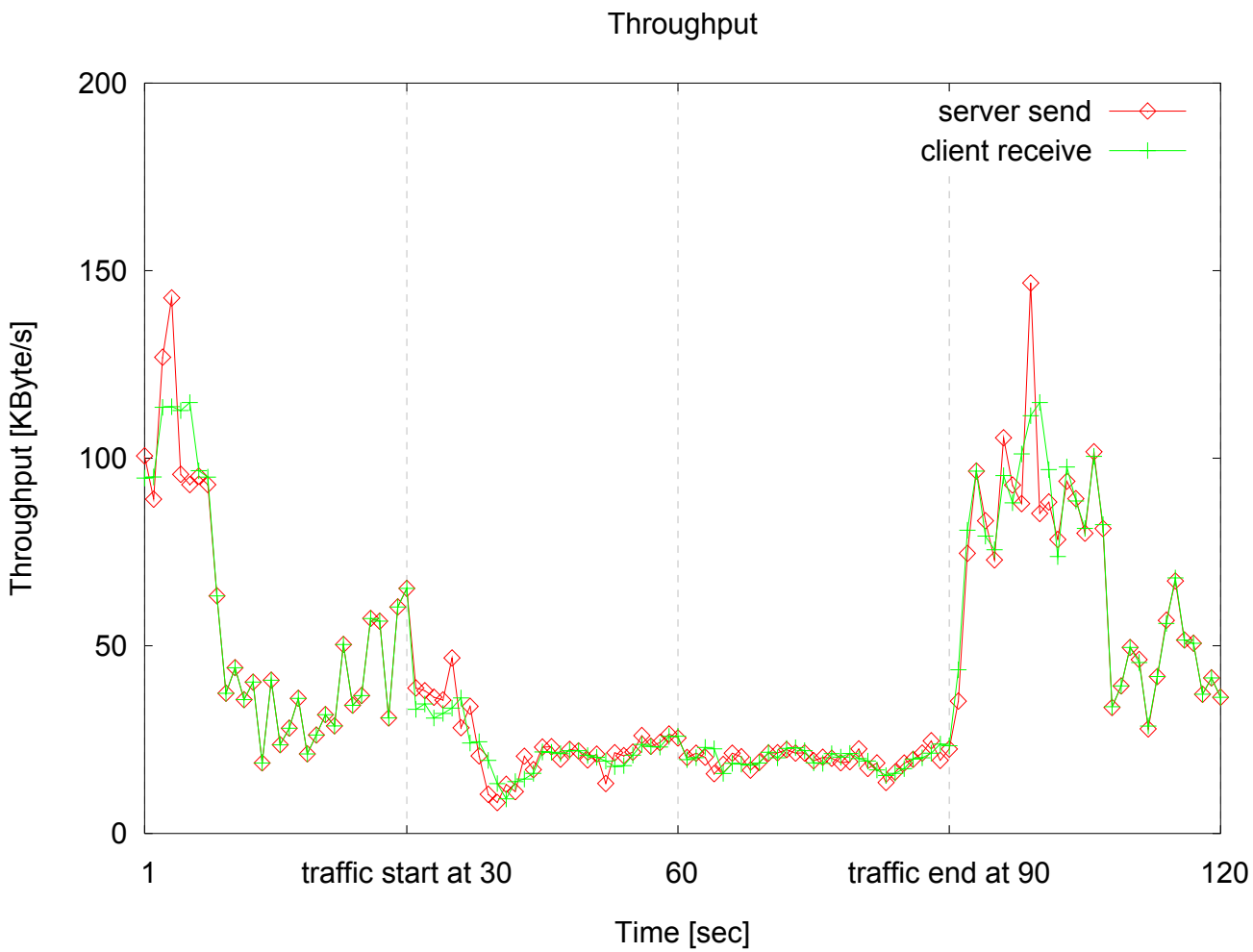
Streaming Video: RealPlayer



Streaming Video: Windows Media Player



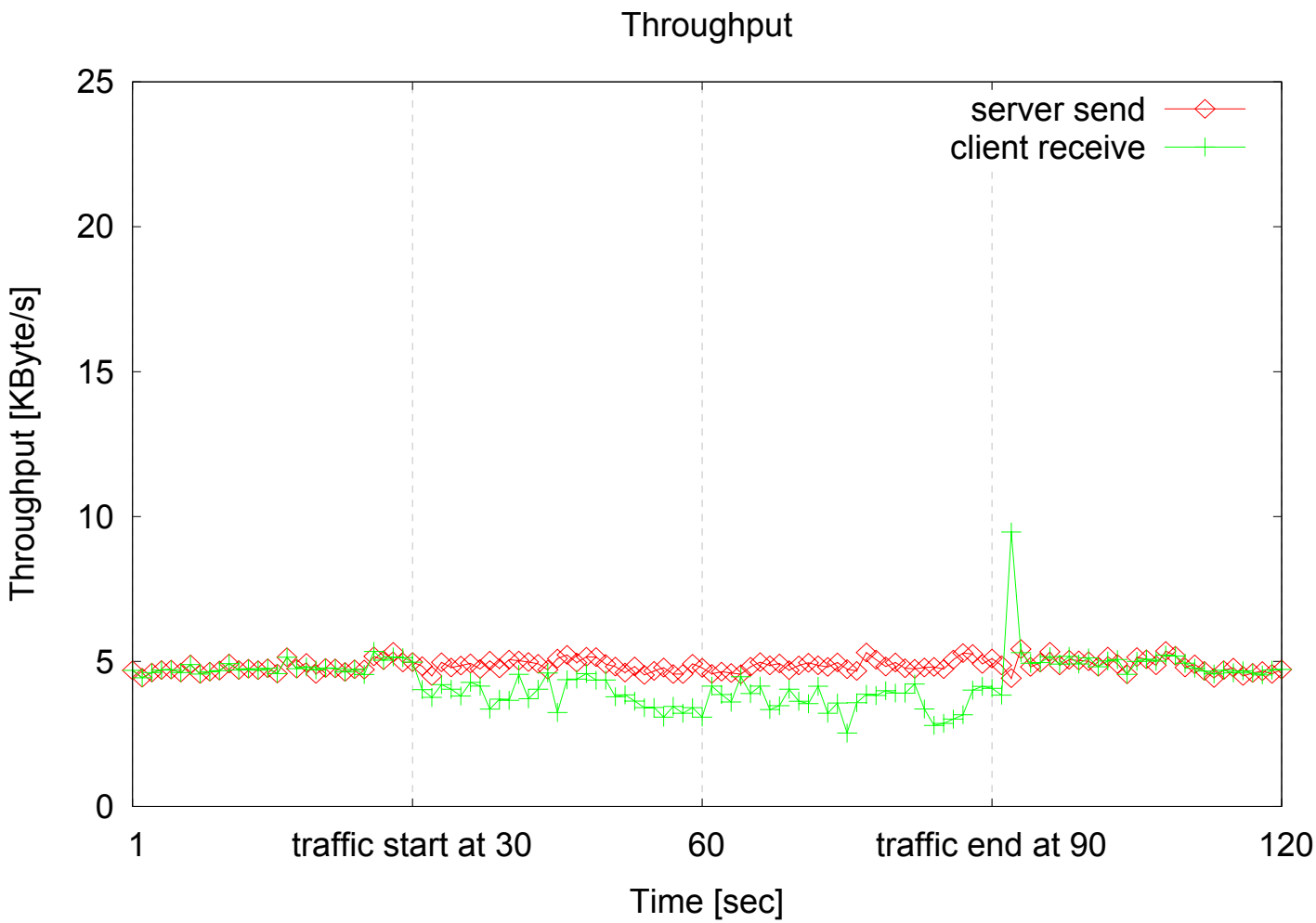
Streaming Video: Quicktime



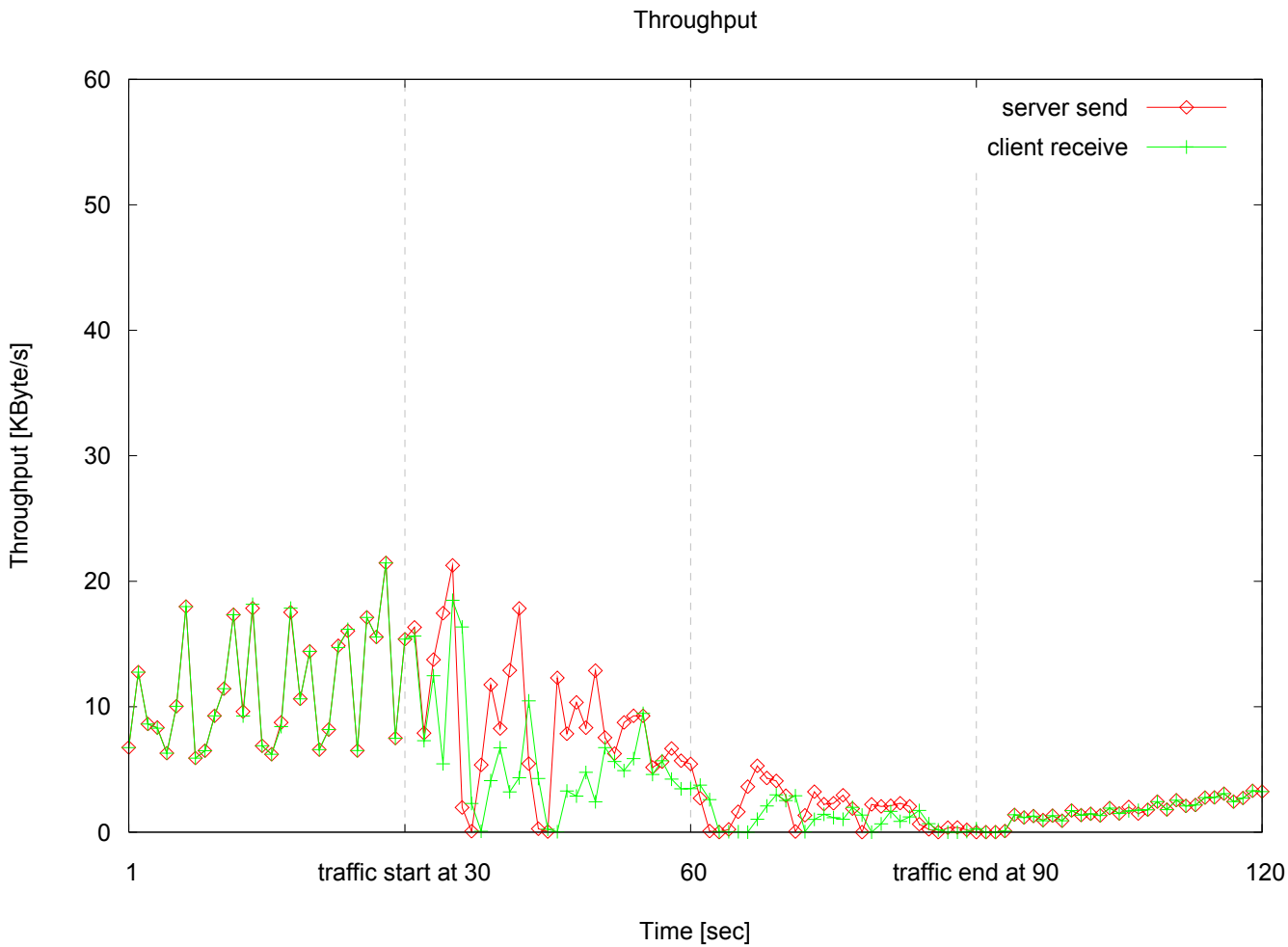
VoIP: MSN


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]

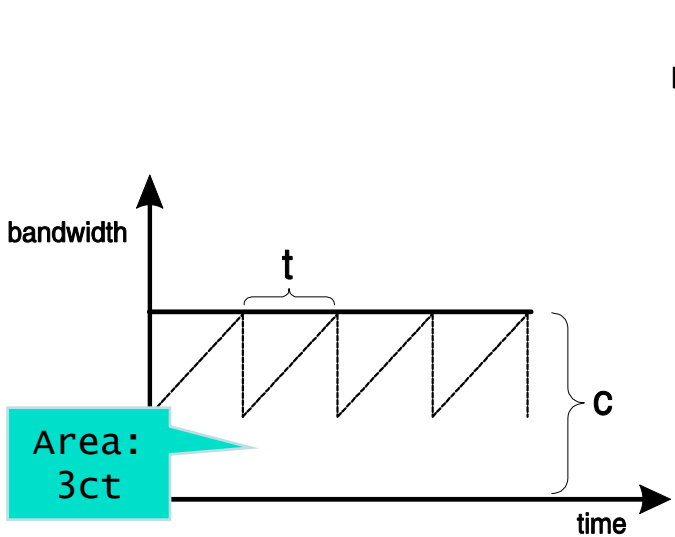
Even TCP without UDP doesn't work too well...

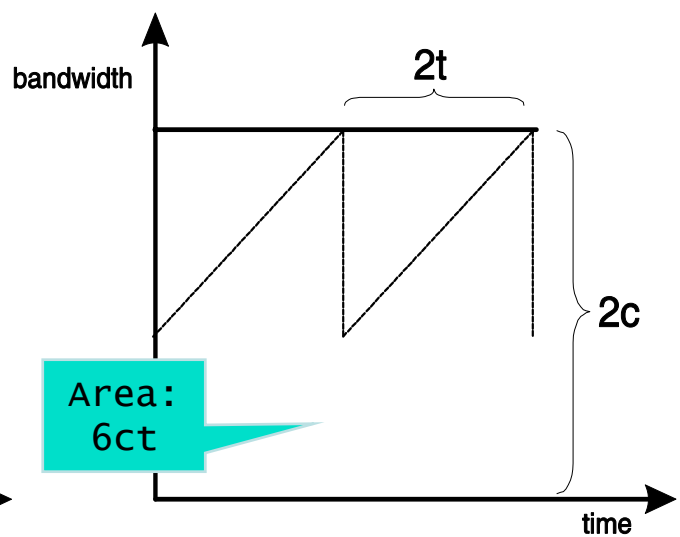
- TCP over "long fat pipes": large bandwidth*delay product
  - long time to reach equilibrium, MD = problematic!
  - From RFC 3649 (HighSpeed RFC, Experimental):

> 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

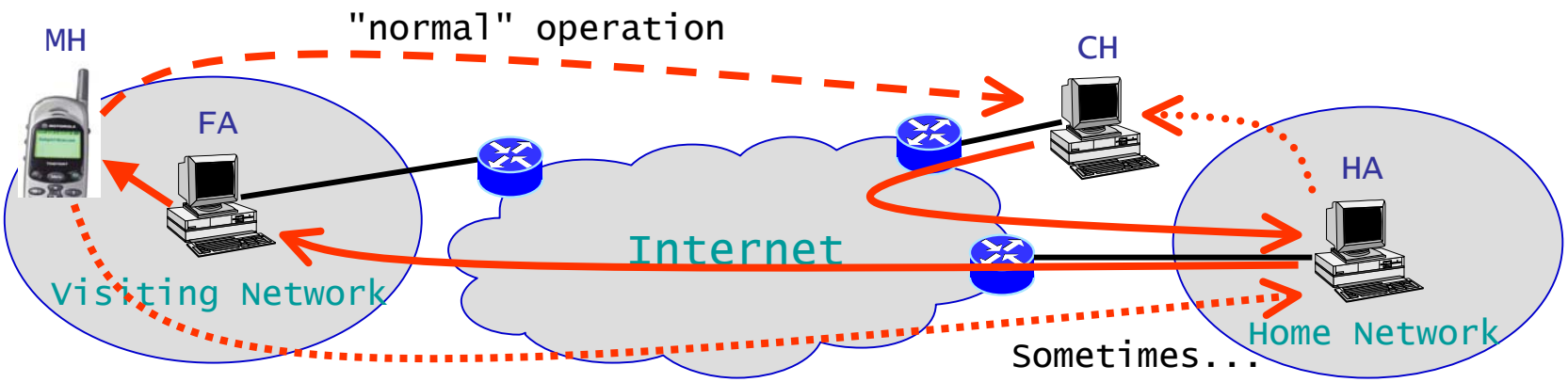
But: longer convergence time





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

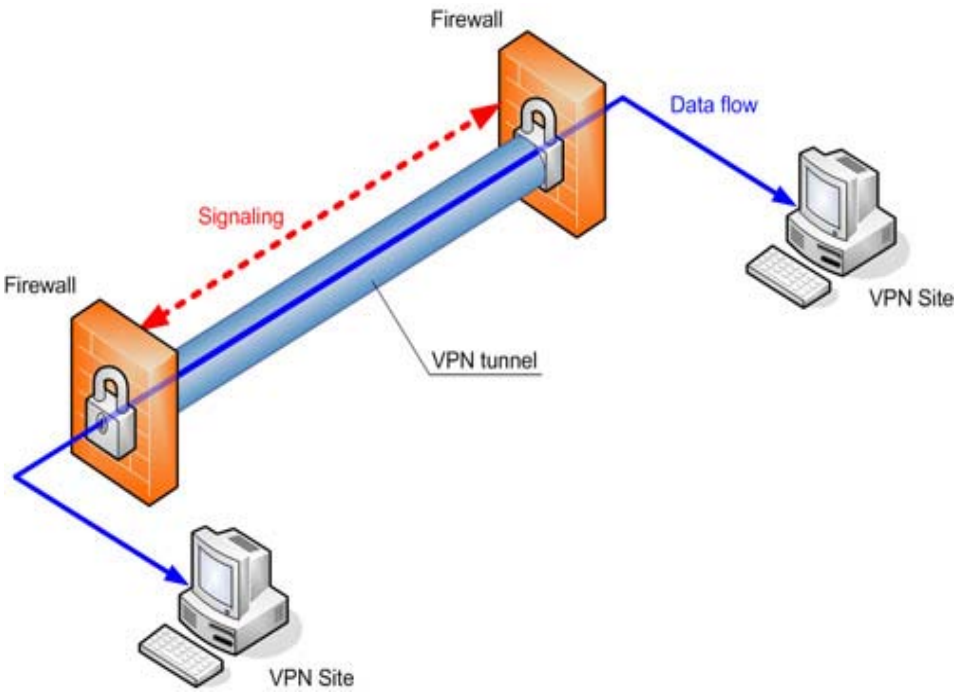


Middlebox End-to-end Performance Enhancements for VPNs

Scenario and Tools

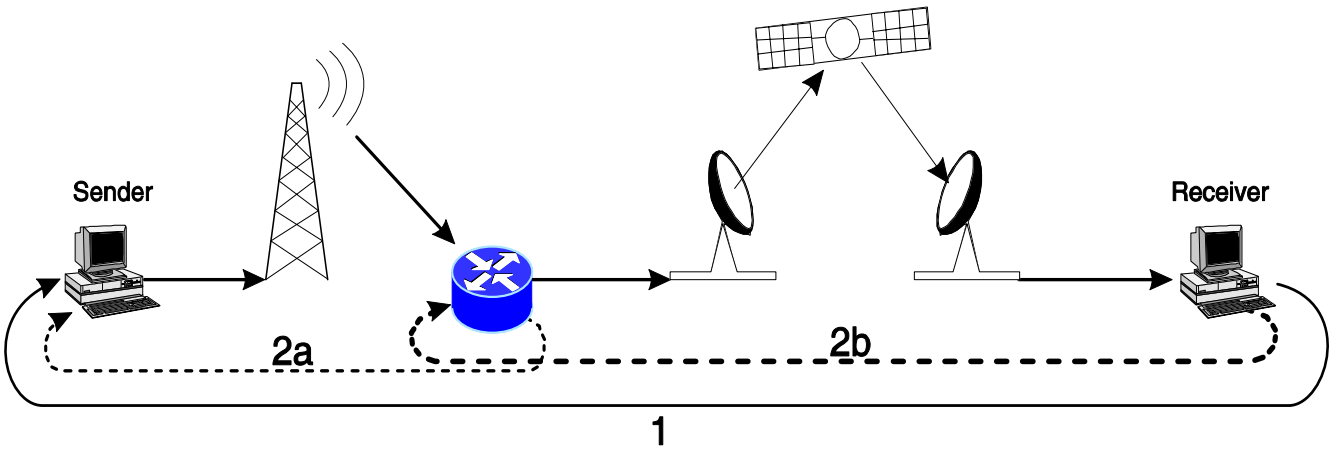
The MEP-VPN usage scenario

- We know that there is a phion device on both sides of the tunnel
- These devices can communicate (feedback about available bandwidth)
- Dynamic traffic control



- Possible to cleverly manipulate control loops in the network; "enforce" congestion control for non-conformant applications
- Goals: greater reliability, enhanced performance

Common Performance Enhancing Proxies (PEPs)

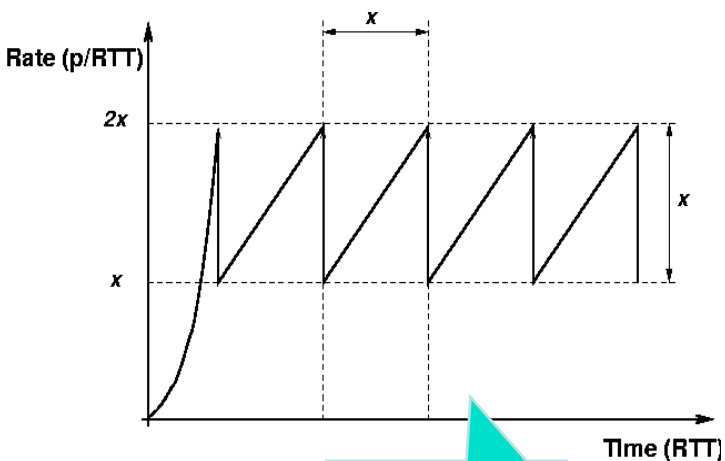


- Figure shows several combined problems
  - Satellite: Long delay, high capacity
  - Wireless: frequent packet loss from corruption
- Example: split connection approach: 2a / 2b instead of control loop 1
  - Many more sophisticated possibilities - e.g. Snoop TCP: monitor + buffer; in case of loss, suppress DupACKs and retransmit from local buffer
- Note: different scenario - only one PEP!
  - Cannot use congestion feedback ⇒ cannot use the things to come...

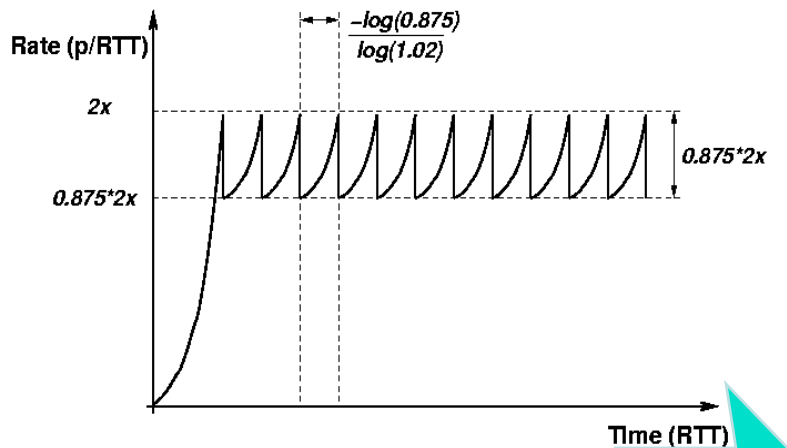
The things we can do

- Many research proposals for enhancing TCP congestion control

- Scalable TCP: increase/decrease functions changed
  - cwnd := cwnd + 0.01 for each ack received while not in loss recovery
  - cwnd := 0.875 * cwnd on each loss event
    (probing times proportional to rtt but not rate)



Standard TCP



Scalabe TCP

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

Better Congestion Control, contd.

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(cwnd) and b(cwnd), which also depend on loss ratio
  - less drastic in high bandwidth environments with little loss ***only***
  - **Significant step: ensures TCP-friendliness and better performance**
- 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
- Many others! e.g. FAST (media appearances!), BIC, CUBIC, XCP, ..

Stream Control Transmission Protocol (SCTP)

- TCP, UDP do not satisfy all application needs
- SCTP evolved from work on IP telephony signaling
  - Proposed IETF standard (RFC 2960)
  - Like TCP, it provides reliable, full-duplex connections
  - Unlike TCP and UDP, it offers new delivery options that are particularly desirable for telephony signaling and multimedia applications
- TCP + features
  - Congestion control similar; some optional mechanisms mandatory
  - Two basic types of enhancements:
    - performance
    - robustness

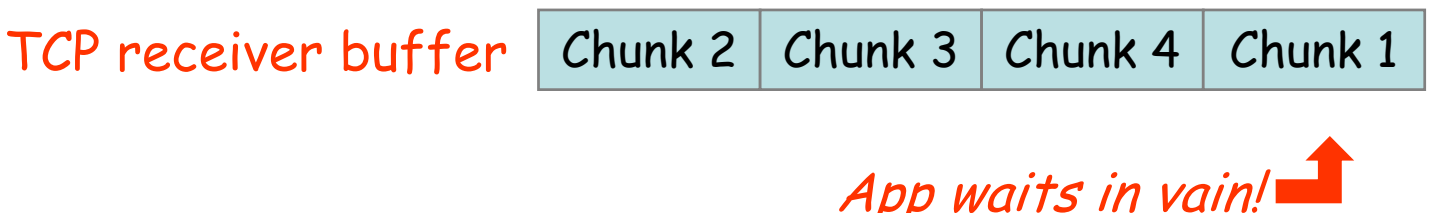
Overview of services and features

Services/Features	SCTP	TCP	UDP
Full-duplex data transmission	yes	yes	yes
Connection-oriented	yes	yes	no
Reliable data transfer	yes	yes	no
Unreliable data transfer	yes	no	yes
Partially reliable data transfer	yes	no	no
Ordered data delivery	yes	yes	no
Unordered data delivery	yes	no	yes
Flow and Congestion Control	yes	yes	no
ECN support	yes	yes	no
Selective acks	yes	yes	no
Preservation of message boundaries	yes	no	yes
PMTUD	yes	yes	no
Application data fragmentation	yes	yes	no
Multistreaming	yes	no	no
Multihoming	yes	no	no
Protection agains SYN flooding attack	yes	no	n/a
Half-closed connections	no	yes	n/a

SoA TCP
+ Extras

Performance enhancements

- Decoupling of reliable and ordered delivery
  - Unordered delivery: eliminate head-of-line blocking delay

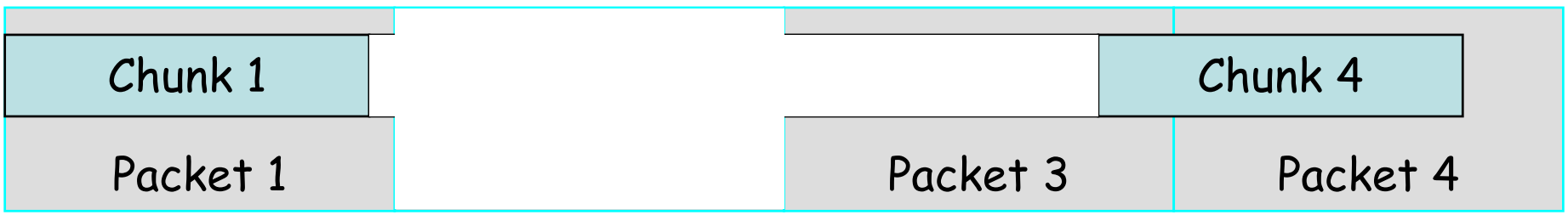


- Application Level Framing

- Support for multiple data streams (per-stream ordered delivery)
  - Stream sequence number (SSN) preserves order *within* streams
  - no order preserved *between* streams
  - per-stream flow control, per-association congestion control

Application Level Framing

- TCP: byte stream oriented protocol
- Application may want logical data units ("chunks")
- Byte stream inefficient when packets are lost



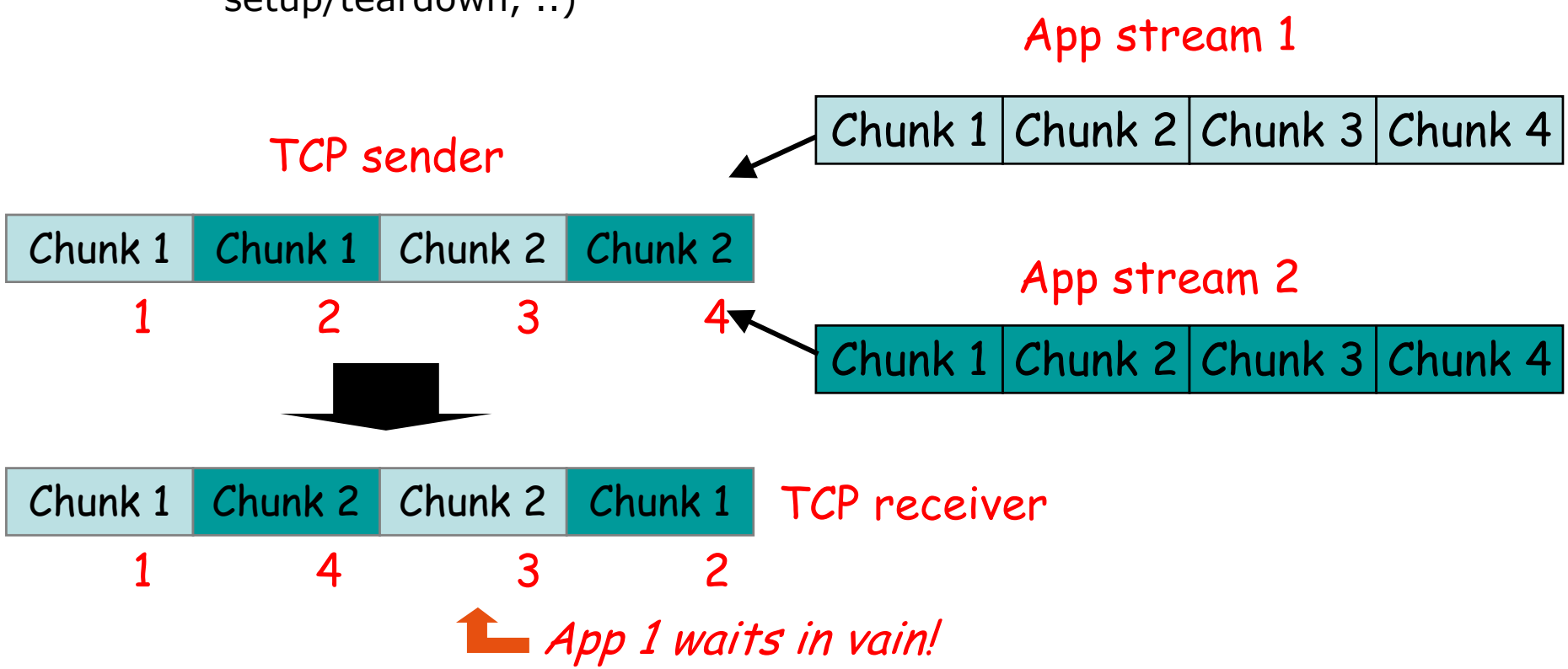
- ALF: app chooses packet size = chunk size

  packet 2 lost: no unnecessary data in packet 1,
  use chunks 3 and 4 before retrans. 2 arrives

- 1 ADU (Application Data Unit) = multiple chunks -> ALF still more efficient!

Multiple Data Streams

- Application may use multiple logical data streams
  - e.g. pictures in a web browser
- Common solution: multiple TCP connections
  - separate flow / congestion control, overhead (connection setup/teardown, ..)



Multihoming

- ...at the transport layer! (i.e. transparent for apps, such as FTP)
- TCP connection ⇔ SCTP association
  - 2 IP addresses, 2 port numbers ⇔ 2 sets of IP addresses, 2 port numbers
- Goal: robustness
  - automatically switch hosts upon failure
  - eliminates effect of long routing reconvergence time
- TCP: no guarantee for "keepalive" messages when connection idle
- SCTP monitors each destination's reachability via ACKs of
  - data chunks
  - heartbeat chunks
- Note: SCTP uses multihoming for redundancy, not for load balancing!

Conclusion

- Researchers have proposed many things
  - Better-than-TCP congestion control for
    - Noisy links
    - Asymmetric connections
    - High-speed links (with large bandwidth * delay product)
  - SCTP, to selectively disable TCP features
  - Intelligent queuing mechanisms for controlling UDP flows
- But they are not used
  - End-to-end deployment is tricky
  - often: chicken - egg problem
- In the MEP-VPN scenario, we will use them!
- Where's the research?
- Research proposals consider end-to-end scenario, so changes are needed...

Questions?