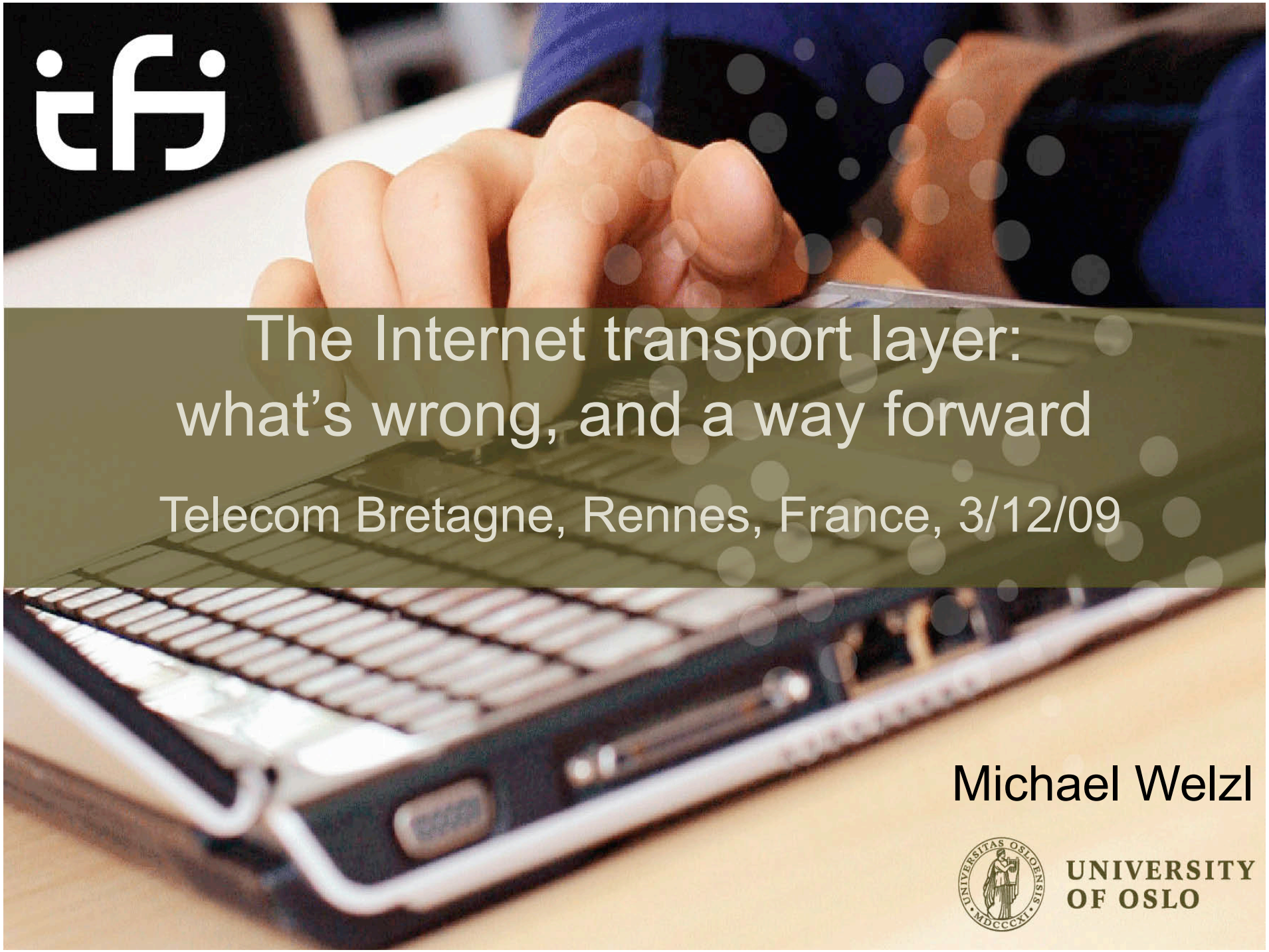


The Internet transport layer: what's wrong, and a way forward

Telecom Bretagne, Rennes, France, 3/12/09

Michael Welzl

UNIVERSITY OF OSLO

Is there really something wrong?

- It works
  - and has been working for a long time now
  - The Internet is very successful, with continuous growth, so why do I say that something's wrong?

- Success = quality?
  - MS Windows is also very successful
  - Opinions diverge…

What do I claim to be wrong?

- It can't really be improved!

- Internet transport layer = TCP, UDP
  - TCP: RFC 791, 1981. UDP: RFC 768, 1980.

- Probably only two truly significant changes:
  1. Addition of congestion control to TCP: 1988
  2. Change of default TCP CC. in Linux to BIC: 2004
     (a bit later: CUBIC) ... not IETF-approved!

IETF has developed much more

- Getting deployed:
  - Many, many TCP bug fixes
- Hardly getting deployed:
  - New protocols: UDP-Lite, SCTP, DCCP
- Newer things - can't evaluate deployment yet (but don't want this to end up "in the red" !)
  - PCN, LEDBAT, MPTCP…

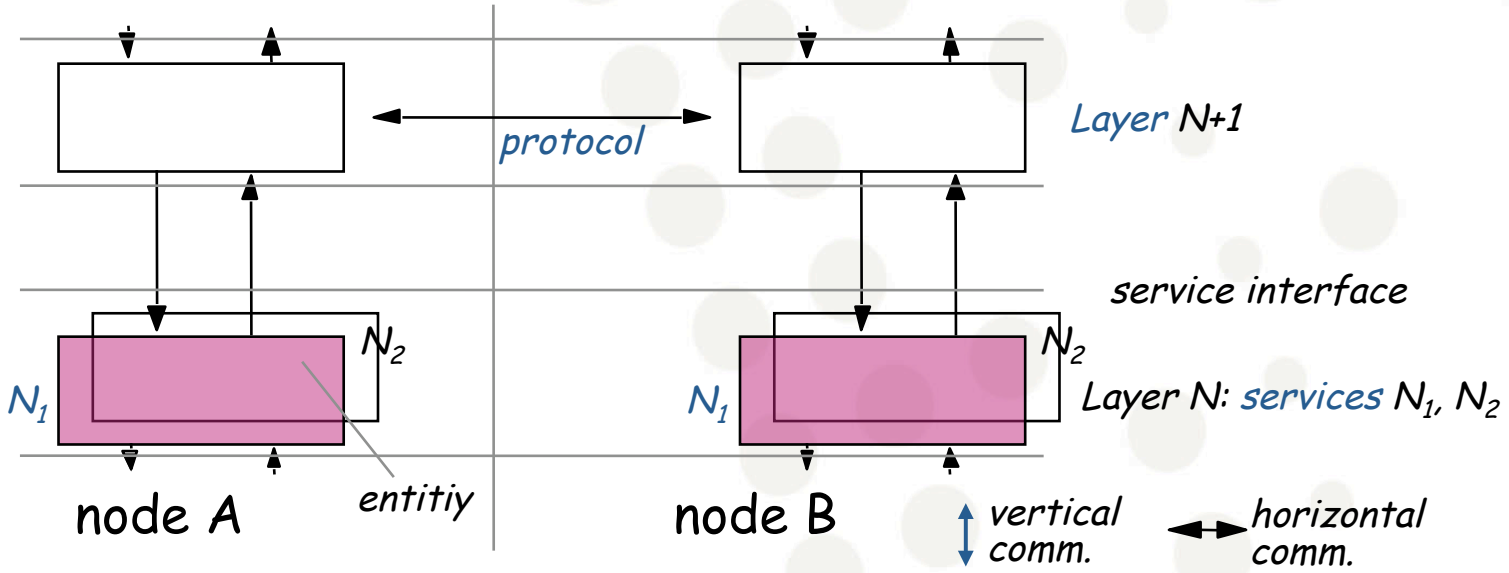
There's an underlying design flaw

(Just my opinion of course)

- Let's talk about OSI
  - OSI failed, DoD model (TCP/IP) was successful, so is OSI even worth thinking about?
  - Again: success = quality?
  - Failure of OSI as a protocol suite doesn't mean that there were no good ideas there

Note: sorry if I'm misrepresenting history - I'm not an Internet historian, and I wasn't "there" when it all happened

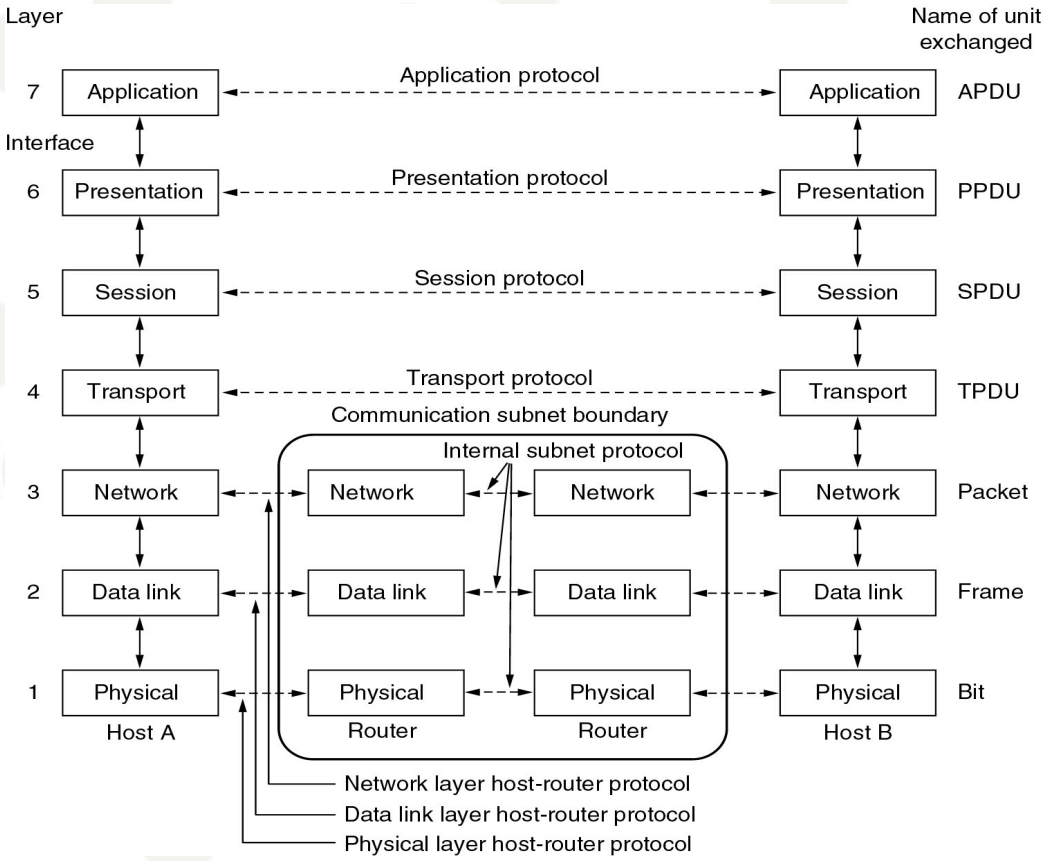
OSI



- Key idea: <u>abstraction</u>
  - Layers merely provide a service
  - Inner operation + lower layers hidden ➔ <u>could be replaced!</u>

OSI vs. the Internet

- Transport layer <u>should</u> be easy to change
- Internet abstraction: socket interface
  - TCP stream
  - UDP datagrams
- Service = what these protocols provide
  - Not very abstract



Source: A. Tanenbaum, Computer Networks

The Internet vs. OSI

- The Internet was not designed for security, lots of problems with this
  - so there's a tendency to disable/block everything that looks "strange" in middleboxes / firewalls
  - TCP, UDP, and special applications, i.e. port numbers, are considered acceptable
    - everything else is "strange"
  - "Deep Packet Inspection" is very much non-OSI
    - and against Internet design: "be liberal in what you accept"
    - But what can we do, sue maintainers / developers of these boxes?

Well, maybe we can! thanks to network neutrality!

That’s it. This ends the talk.

Let’s all file a lawsuit together.
Problem solved.

A way forward

Pragmatic incentive view

- I believe that most Internet deployment failures (yes also QoS) are at least partially due to misaligned incentives
  - We should no longer develop technology without considering this

- I'm not the first one to say this:

  David D. Clark, John Wroclawski, Karen R. Sollins, Robert Braden: "Tussle in cyberspace: defining tomorrow's internet", SIGCOMM 2002

  - Let's apply these principles to the transport layer...

The transport tussle

1. Application designers
   - want to get best performance with minimal effort
     - Note: difference between updating an already working application and writing a new one from scratch
   - make use of a protocol which is now only available in 1% of the world: usually not worth it
     - Note for commercial applications: programming effort = time = money
   - Future: if things change, we can still update our application

The transport tussle /2

2. OS developers
   - want to get best performance with minimal risk
   - e.g. Linux: it seems that whatever makes the OS work better without reducing stability is welcome
   - supporting a protocol which might be used one day is not a big risk, maybe worth it (in Linux, even protocol designers do the work)

- Note: if only this group says “yes”, not much might happen.

The transport tussle /3

3. Designers (and maintainers – but let's ignore them for now and assume that they use "system defaults") of middleboxes / firewalls
   - Devices / software often promise "security and good network performance"
   - Whatever is unknown can be a security risk
   - But: if blocking something notably degrades performance, customers won't like that
     - might not block it by default

Analyzing success stories

- TCP “bug fixes”
  - in accordance with originally planned behavior
  - installing in OS (group 2) yields a direct benefit for group 2 (and group 1)

- (CU)BIC congestion control as default in Linux TCP
  - not even a standard! but a major press release + available code, written by the designers
  - installing in OS (group 2 only) yielded a direct benefit for group 2 (and group 1)

How to accommodate the tussle?

- We are talking about people here; no hard facts, nothing is set in stone
  - People can change their minds
  - Group 3 is often seen as unchangeable; I don't believe in this
  - **Main "message" of this talk: we should take this tussle serious, and develop suitable technology!**
  - Shorter-term and longer-term plans possible
    - Three ideas for getting existing protocols deployed follow (long-term and short-term)

Idea 1, long term: just imagine...

...that we'd have a more abstract transport API.

1. Applications say...
   - what kind of service they prefer
   - what kind of traffic they will generate
2. Using its resources (protocols, signaling with the inner network, ...), the transport layer does its best (still best effort!) to provide a good service
   - Could try a new protocol, and give up in case of failure
   - Could maybe also answer: "hey, you're even getting a guarantee here!"

Idea 1, cont'd

- I believe that such an API + transport operation below is key for solving the problem
  - Again, I'm not the only one thinking this way...

    Bryan Ford and Janardhan Iyengar: "Breaking Up the Transport Logjam", HotNets-VII, October 2008. http://www.brynosaurus.com/pub/net/logjam.pdf

    Michael Welzl: "A Case for Middleware to Enable Advanced Internet Services", NGNM'04 workshop, co-located with Networking 2004, Athens, Greece, 14 May, 2004 http://heim.ifi.uio.no/~michawe/research/publications/ngnm04.pdf

- But this would probably have the same (intermediate?) deployment problems
  - so, not enough

2. Make protocols more attractive

- Example: DCCP, group 1 point of view
  - + Suitable congestion control for unreliable real-time applications (e.g. streaming media)
  - + No need to do the work in applications above UDP
  - + Good performance (cc. in the OS, where it belongs), TCP-friendly, support for ECN
  - – New API to use
  - – New protocol, not yet available in all OS's, many middleboxes will drop it

Congestion control trade-off

(selfish single-flow view)

\+ reduced loss

— necessary to adapt rate

– Use sender buffer, drain it with varying rate

– Change encoding

Trade-off: sender buffer size (=delay) vs. frequency of encoding changes

VoIP, Games

Videoconf.

Streaming Media

Sweet spot?

Delay sensitive

Delay insensitive

Is TCP the ideal protocol for one-way streaming media?

- Remember: we're at the "buffering" side of the spectrum
  - Buffers (delay) don't matter
  - User perception studies of adaptive multimedia apps have shown that users dislike permanent encoding changes (big surprise :-))

  ⇒ no need for a smooth rate!
- Little loss case: TCP retransmissions won't hurt
- Heavy loss case:
- DCCP: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10…
- TCP: (assume window = 3): 1, 2, 3, 2, 3, 4, 3, 4, 5, 4…
  - Application would detect: 4 out of 10 expected packets arrived
    ⇒ should reduce rate
  - Is receiving 1, 4, 7, 10 instead of 1, 2, 3, 4 really such a big benefit?
    - Or is it just a matter of properly reacting?
    - See TCP usage in RealPlayer, MediaPlayer, YouTube, Andreas Petlund's Ph.D. thesis

Idea 2, cont'd

- How can we make DCCP more attractive?
  - "TCP-friendliness" is maybe not very interesting from the user point of view:
    - preserve network stability
    - ensure that other flows (other applications of the same user, or other users) get a fair bandwidth share
  - TCP-friendliness also means 75% link utilization if we're the only application
    (or more if buffers (➔ and delay) are large)

Idea 2, cont'd: MulTFRC

Dragana Damjanovic, Michael Welzl: "MulTFRC: Providing Weighted Fairness for Multimedia Applications (and others too!)", ACM Computer Communication Review 39(3), July 2009.

- Like TCP-friendly Rate Control (TFRC), but weighted: can act like N flows, where N is a positive rational number
  - 3 flows: 90% link utilization. 6 flows: 95%

    Utilization = 100-100/(1+3N) percent. From:
    E. Altman, D. Barman, B. Tuffin, M. Vojnovic: "Parallel TCP Sockets: Simple Model, Throughput and Validation", Infocom 2006.
  - 0 < N < 1 also works just fine
- For DCCP based applications, this means: can saturate links better, can give a knob to users

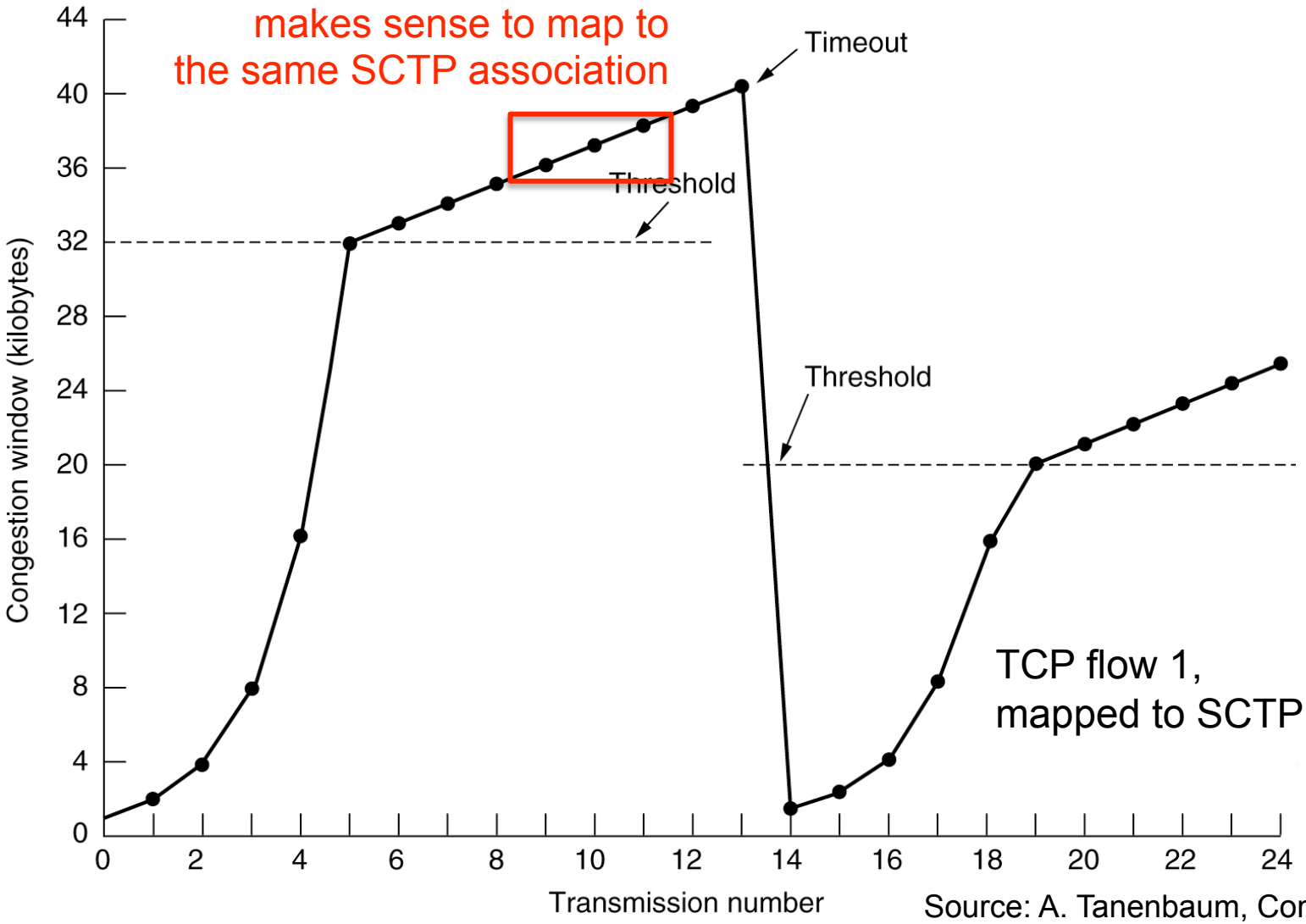
3. Beneficial Transparent Deployment

- For a new protocol, first show that there can be a benefit from transparently deploying it
  - in the OS; involves only group 2
  - always ensure fallback, no disadvantage from trying the new protocol; could eventually give more and more people from group 3 a reason to say “yes”
  - once group 2 and group 3 have it, it makes sense for group 1 to use it ➔ full benefit!

Idea 3, cont'd: SCTP example

- SCTP is already (somewhat?) attractive
  - resilience can improve if used transparently (automatically use multihoming)

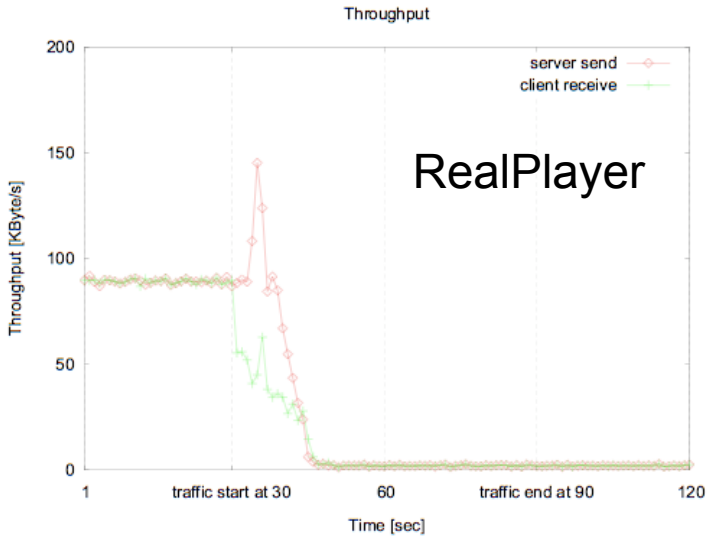
- Can get more benefits out of transparent usage: using multi-streaming
  - map short TCP connections onto long SCTP association, exploit large congestion window IFF this yields a benefit
  - Sorry, no diagram; one of my master students is working on this, he says it works... we plan to have a paper soon

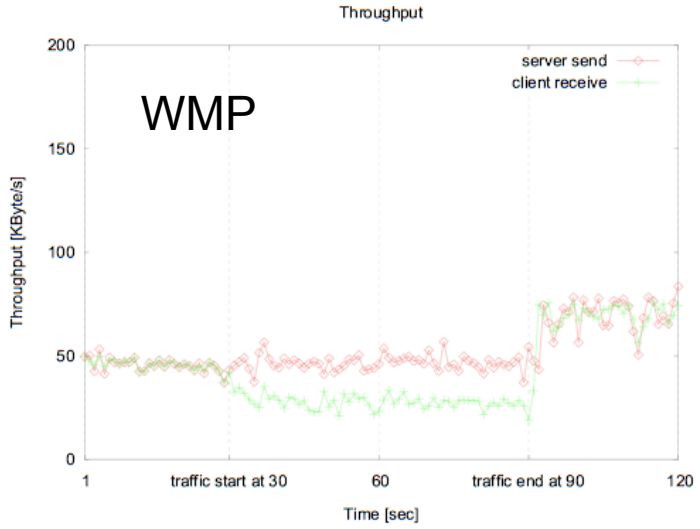


Source: A. Tanenbaum, Computer Networks

Idea 3, cont'd: DCCP example

- Can we transparently use DCCP such that a noticable benefit for applications is attained?
  - enforce appropriate behavior for applications that don't have no, or no "good" congestion control





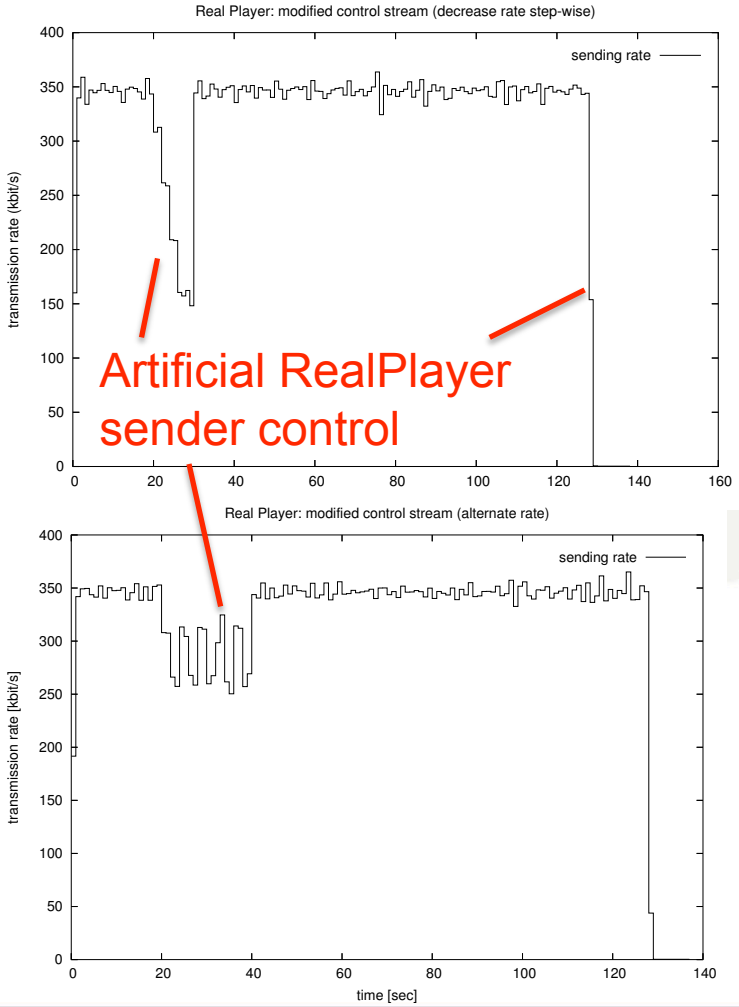
3: Enforcing congestion control

- Various issues:
  - Introduce + control extra buffer close to sender
  - Control sender (explicit or implicit feedback)
  - Get feedback from receiver: maybe less signaling than per-packet ACKs

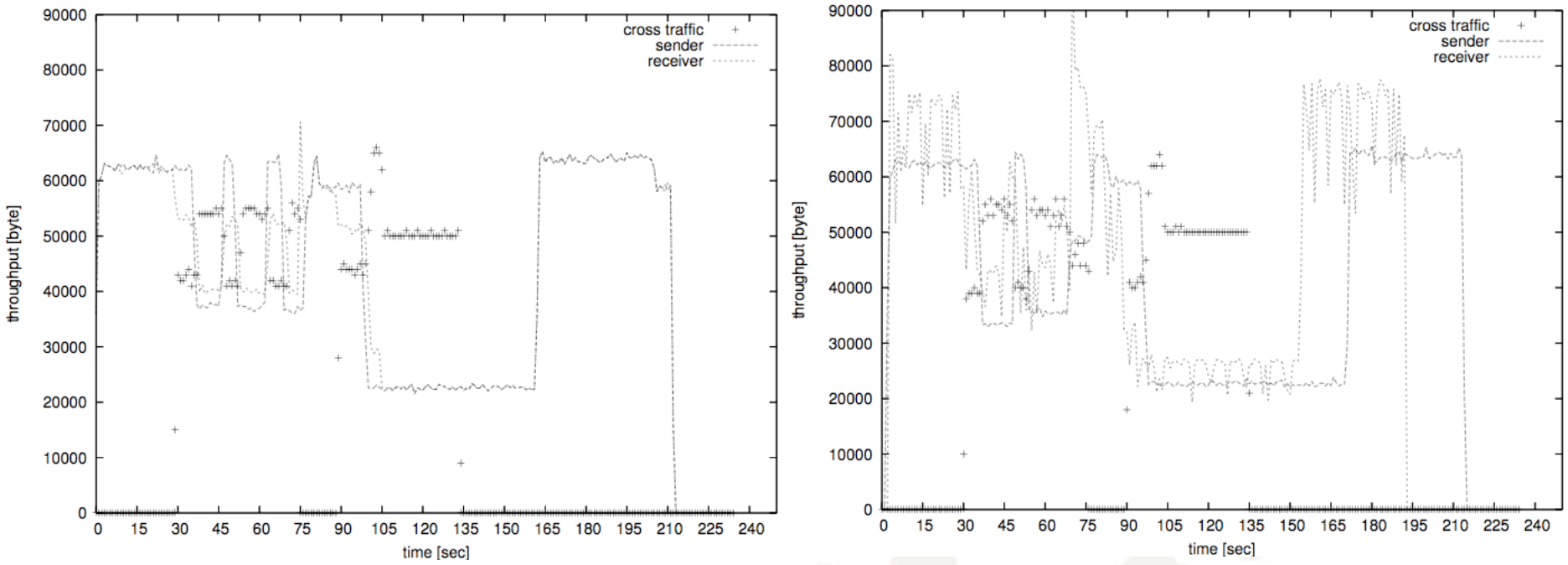
- Solving this is hard, but feasible

  [Sven Hessler, "Protection of Data Networks by Enforcing Congestion Control on UDP Flows", Ph.D. thesis, October 2008, TU Darmstadt]

  Source of diagrams on previous, this, and next slide.



3: Enforcing congestion control /2



RealPlayer without (left) and with(right) congestion-enforcing element

No explicit-feedback sender control

- Still open question: can we get enough benefit for applications?

Conclusion

- I repeat: **main "message" of this talk: we should take this tussle serious, and develop suitable technology!**
  - **Secondary message: also consider aligning existing technology with it**

- There's a lot of work to be done
  - measure what middleboxes do, evaluate encapsulation variants (everything-over-UDP?), connection setup (meta-syn, or multiple SYNs ("happy eyeballs"), or TCP SYN with a special flag?)... gets even more difficult (= interesting!) when routers are involved

- Let's avoid repeating past mistakes over and over again, and really improve the Internet

A funding view

- Some time ago, it was said, often, everywhere:

"We need a clean-slate design"

1. don't care about the Internet, do something new
2. think about gradually moving to the new thing

- A lot of money has gone into 1)
  - It's time to ask for money for 2) !

Thank you

Questions?