What's wrong?

- <u>It can't be changed.</u>
- Internet transport layer = TCP (1981), UDP (1980)
  - Service = what these protocols provide; neither matches app. requirements nor infrastructure capabilities
    (cf.: the tons of papers on app-x-over-... and tcp-over-y)
- Probably only two truly significant (noticeable for users) 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!

2

UNIVERSITY OF OSLO

IETF has developed much more

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

3

UNIVERSITY OF OSLO

SCTP and DCCP in a nutshell

- SCTP: TCP++ ... mostly by removing "features"!
  - TCP without stream semantics, requirements for ordered or reliable delivery
  - and a few features added, e.g. multistreaming (ordered delivery *within* streams only) and multihoming
- DCCP: congestion control for real-time (multimedia) applications
  - various CCID specs define different CC behaviors
  - e.g. TFRC for smoother rate (less jitter for real-time apps)

4

UNIVERSITY OF OSLO

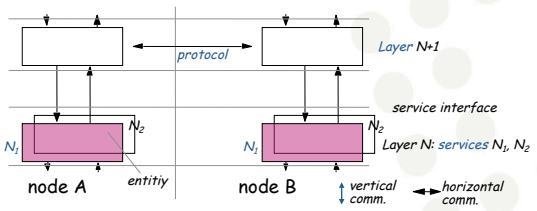
What's wrong? (cont'd)

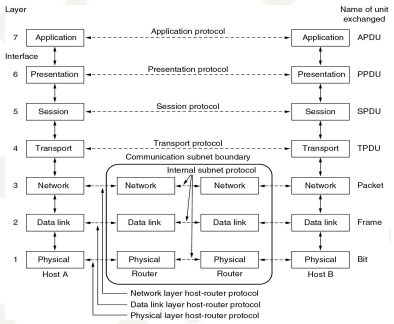
- Internet was not designed for security
  - hence, tendency to disable/block everything that looks "strange"
  - TCP, UDP, and special applications, i.e. port numbers, are considered acceptable; everything else is "strange"
    - ➔ Application programmers don't use other transport protocols
- Design was supposed to be open...
  - "Be conservative in what you send, liberal in what you accept"
  - Reality is different (Deep Packet Inspection, ..)
- What went wrong?

5

UNIVERSITY OF OSLO

Internet design flaw: no abstraction





Source: A. Tanenbaum, Computer Networks

- OSI had the right idea! :-) ...<u>abstraction.</u>
  - Layers merely provide a service
  - Lower layers + their internal operation hidden ➔ could be replaced
- Transport layer should be <u>especially</u> easy to change!

6

UNIVERSITY OF OSLO

A better Internet transport design

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!"

7

UNIVERSITY OF OSLO

A better Internet transport design /2

- 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
- The problem might not have occurred with this...
  - but this doesn't help us now.
  - so how can we get there?

8

UNIVERSITY OF OSLO

A way forward

Pragmatic incentive view

- It seems that most Internet deployment failures (yes also QoS towards end users) 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
   - making 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

11

UNIVERSITY OF OSLO

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)

12

UNIVERSITY OF OSLO

The transport tussle /3

3. Designers 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

13

UNIVERSITY OF OSLO

Remember these groups

1. Application designers
2. OS developers
3. Designers of middleboxes / firewalls

- Each group has "support groups" that share their interests, i.e. no need to explicitly consider them
  1. customers (want a good price)
  2. customers (want a high-performance OS)
  3. device maintainers (might use system defaults)

14

UNIVERSITY OF OSLO

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;
    let us not believe in this (if we do, we're giving up!)
  - Simplification: actually more stakeholders involved...
    but ignoring this simple tussle guarantees failure!
- Main "message" of this talk:
  we should take this tussle serious, and develop suitable technology!

15

UNIVERSITY OF OSLO

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)

16

UNIVERSITY OF OSLO

Suggested research agenda

17

UNIVERSITY OF OSLO

Step 1: Beneficial Transparent Deployment

- Achieve a notable benefit by <u>transparently</u> deploying new protocols
  - 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!

18

UNIVERSITY OF OSLO

Step 2: A new API

- When, because of success with step 1, group 1 begins to really use the new protocols, they will be annoyed by the complex Internet transport API
  - SCTP and DCCP: not just two new protocols, but lots of options that come with them
    - SCTP: draft-ietf-tsvwg-sctpsocket-25, 111 pages
    - DCCP: not even specified, given by implementations; lots of options

➔give them a protocol-independent API (just with transport *services*, not protocols)

19

UNIVERSITY OF OSLO

A lot of work is needed

- Continuously measure protocol availability (middlebox impact)
- Transport protocol negotiation / connection setup (ideas on "Meta-SYN" etc.); must incorporate protocol availability checks
- Show a major benefit from transparent deployment of, e.g., SCTP and DCCP
- Probably necessary to make protocols more attractive to users, e.g. add services to DCCP, better demonstrate benefits of SCTP
- Design the new API, develop the "thing" underneath it, in real code, and demonstrate its benefit to users
- Do these things with real-life code, get it accepted in OSes, and work on IETF standardization

20

UNIVERSITY OF OSLO

Example: Transparent SCTP deployment

Based on:
Florian Niederbacher: "Beneficial gradual deployment of SCTP", Master thesis, University of Innsbruck, February 2010, available from:
http://heim.ifi.uio.no/michawe/teaching/dipls/florian_niederbacher.pdf

21

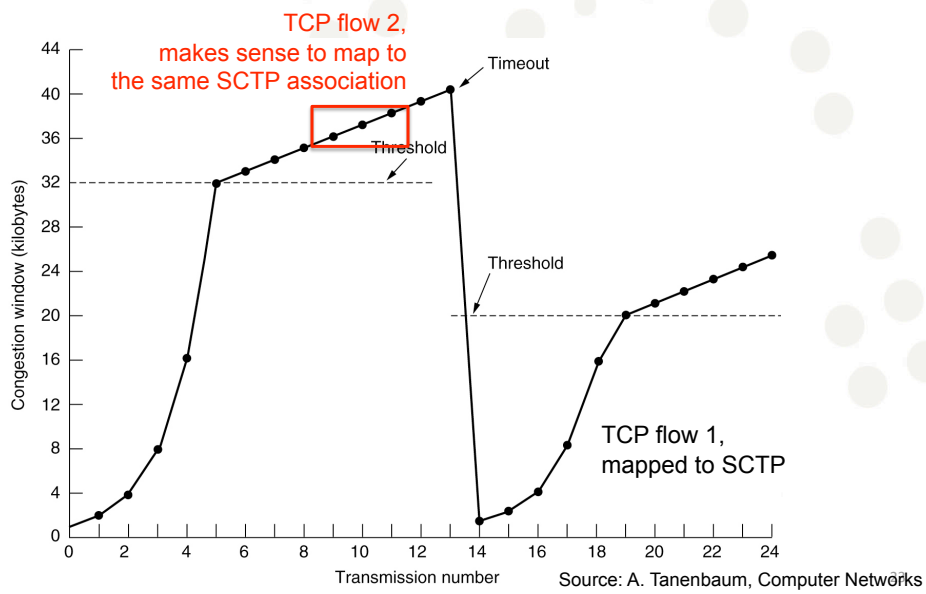
UNIVERSITY OF OSLO

Underlying idea

- 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

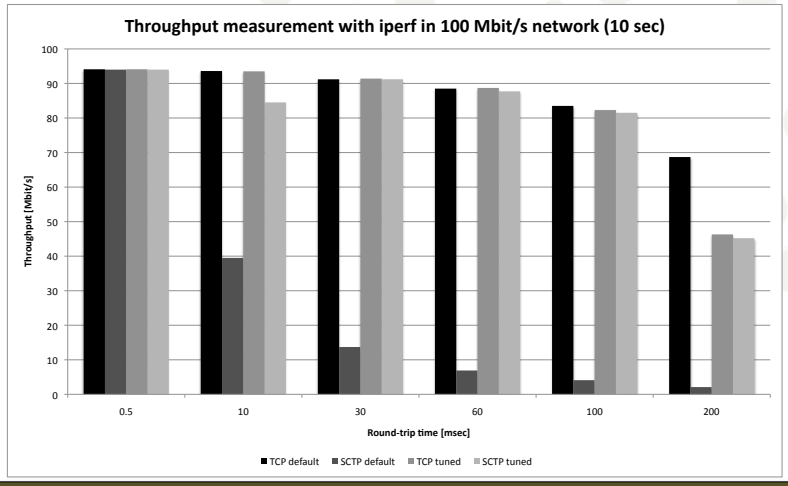
22

UNIVERSITY OF OSLO

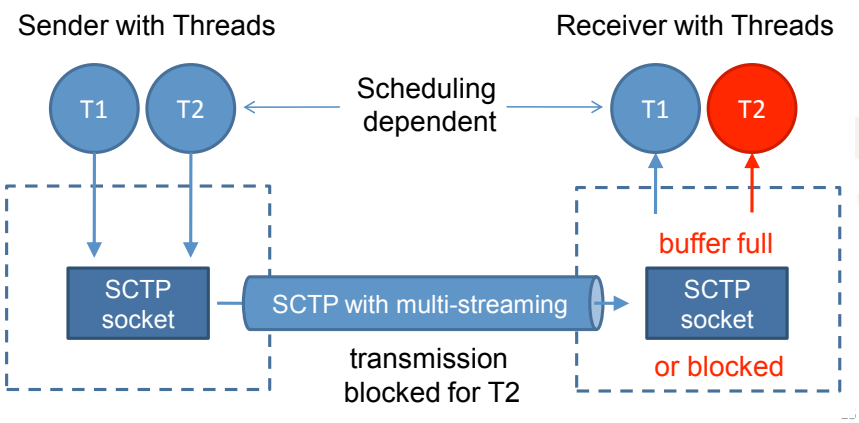


Source: A. Tanenbaum, Computer Networks

Step 1: general performance check

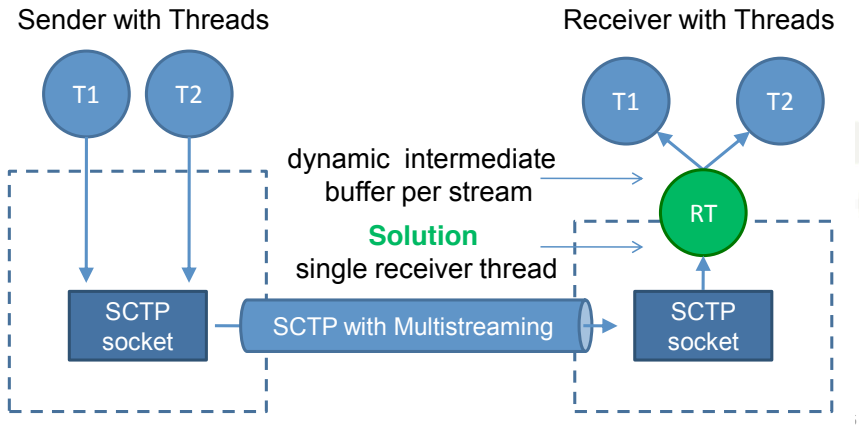


Step 2: implementation



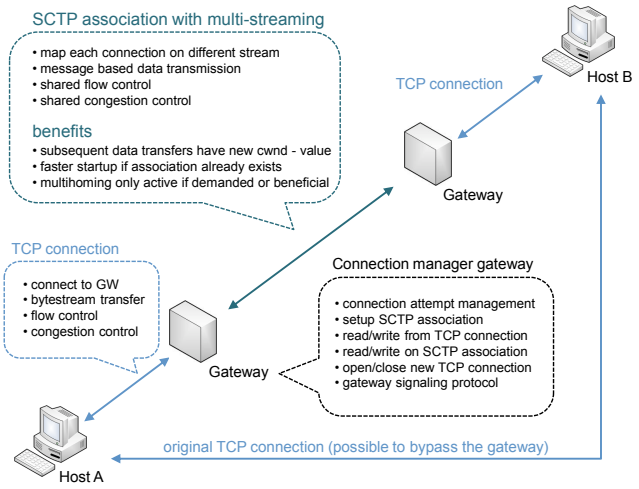
UNIVERSITY OF OSLO

Step 2: implementation /2



UNIVERSITY OF OSLO

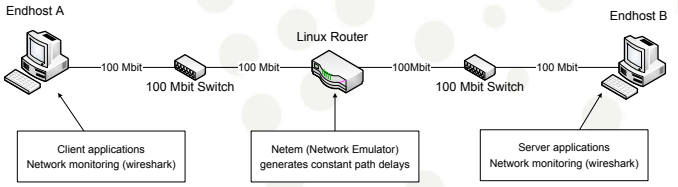
Step 2: implementation /3

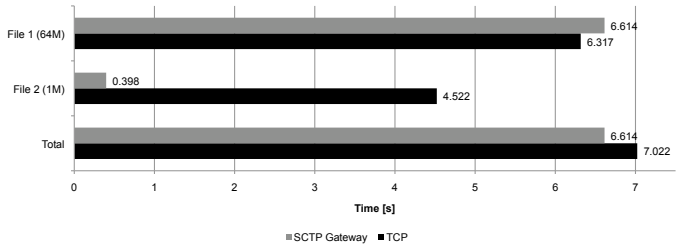


27

UNIVERSITY OF OSLO

Step 3: Test





28

UNIVERSITY OF OSLO

Conclusion from SCTP experiment

- Doing this right is probably worth it... but it's hard
  - kernel implementation required
  - fixes to SCTP required
    - per-stream flow control, improving SCTP performance via auto-buffer-tuning and pluggable congestion control
  - protocol setup / TCP fall-back mechanism required
  - either decide when to map (hard / ugly?) or let SCTP with multiple streams be more aggressive (like MulTCP)

29

UNIVERSITY OF OSLO

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**
- Let's avoid repeating past mistakes over and over again, and <u>really</u> improve the Internet
- <u>A funding view</u>: consider the mantra of "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 get and use some money for 2) !

30

UNIVERSITY OF OSLO

Thank you

Questions?