TCP Meets Mobile Code

Parveen Patel
Jay Lepreau
(*Univ. of Utah*)

David Wetherall
Andrew Whitaker
(*Univ. of Washington*)

The Key Idea

- Transport protocols, such as TCP, need a better upgrade mechanism
- Untrusted mobile code will work!

TCP is a work-in-progress

- A steady stream of TCP extensions and new transport protocols
  - TCP SACK (1996)
  - TCP Connection Migration (2000)
  - ECN and ECN nonce (2001)
  - TCP Nice (2002)
  - TFRC (2000)
  - DCCP (2002)
  - SCTP (2002)
  - …

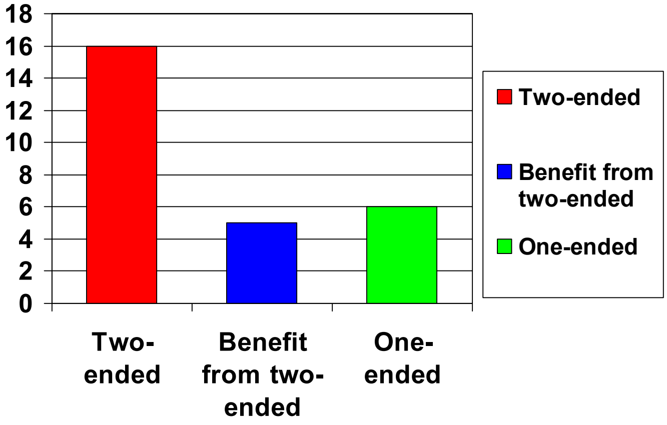
Upgrading TCP takes forever

- Research and simulation
- Prototype
- Standards committee
- Implementation in OS 1
- Implementation in OS 2
- …
- Addition into standard build OS 1
- Addition into standard build OS 2
- …
- Enable by default
- Enable by default on peer

Lousy fallback: one-ended change

- Immediate deployment for self benefit
- Does not always work
  - Can’t exchange new information
- Does not work very well
  - Lose the benefit of cooperation between both ends

Survey of transport extensions



Our Solution: XTCP

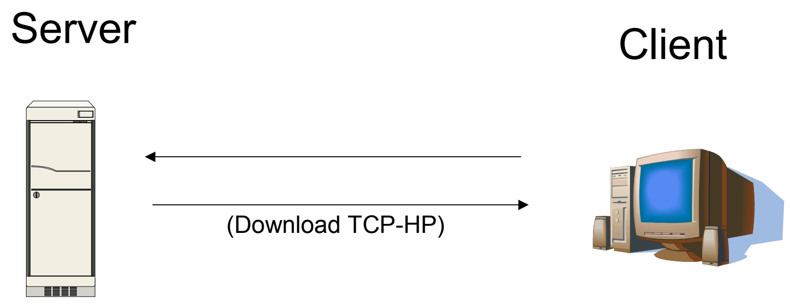
- Connection peers can upgrade each other with new transport protocols using mobile code
- Deployment at one end is all we need !

Upgrading with XTCP is faster

- Research and simulation
- Prototype
- Standards committee
- Implementation to the XTCP API
- Implementation in OS 1
- Implementation in OS 2
- …
- Addition into standard build OS 1
- Addition into standard build OS 2
- …
- Enable by default
- Enable by default on peer

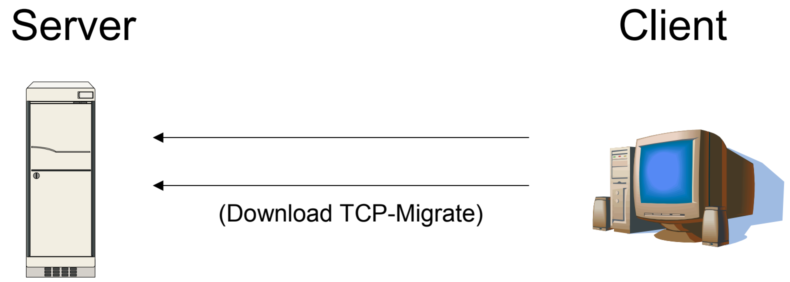
XTCP usage scenario #1

- A web server pushes a “high-performance” TCP to its clients the first time they connect



XTCP usage scenario #2

- A mobile client pushes “TCP connection migration” [MobiCom ’00] to a server to allow itself to move



XTCP usage scenario #3

- A user installs “TCP nice” [OSDI ’02] to support background data transfer

Host A

(With TCP-Nice)

Host B

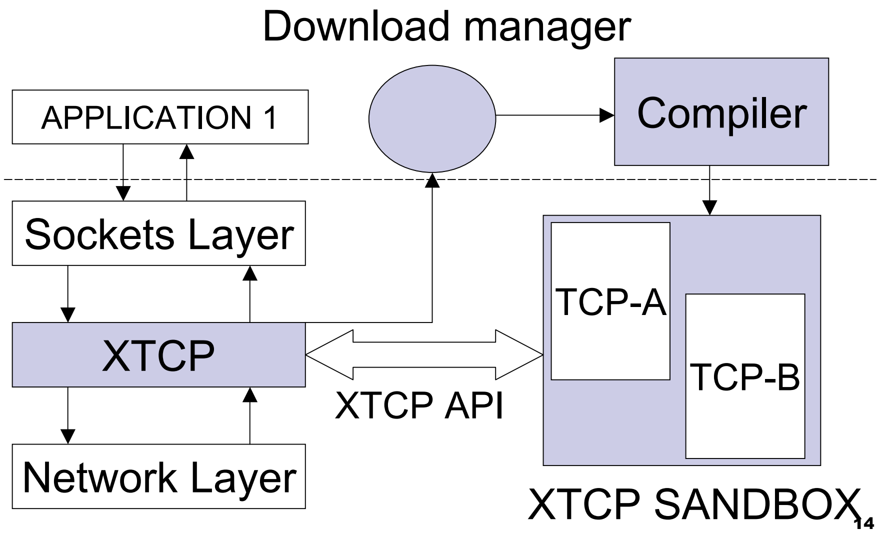
Will it work ?

- XTCP sounds similar to the challenging domain of active networking
- Challenges can be met in this restricted domain

XTCP Challenges

1. Host safety – must isolate and limit resource consumption
2. Network safety – should not compete unfairly or attack other nodes
3. Performance – should not undermine improvement due to extensions

XTCP Design



1. Host safety

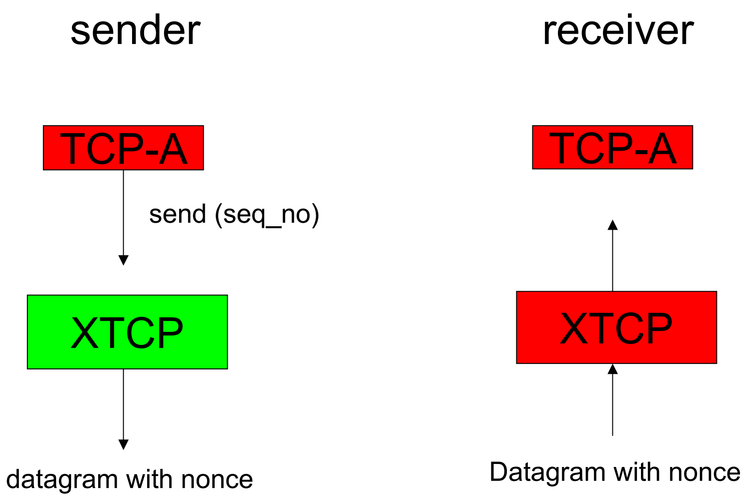
- No shared state between extensions
  - Easy resource accounting
  - Easy termination
- Memory safety: type-safety of Cyclone
- CPU timer-based CPU protection

2. Network safety

- Well-defined notion of network safety
  - TCP-friendliness [RFC 2914]
  - TCP response function is mathematically defined [SIGCOMM ’98]
- Enforcement without trusting transports
  - Adapt ECN nonce mechanism is used for validation [ICNP ’01]

Nonce Mechanism

[unstrusted red, trusted green components]



sender

receiver

🟥 TCP-A

send (seq_no)

🟩 XTCP

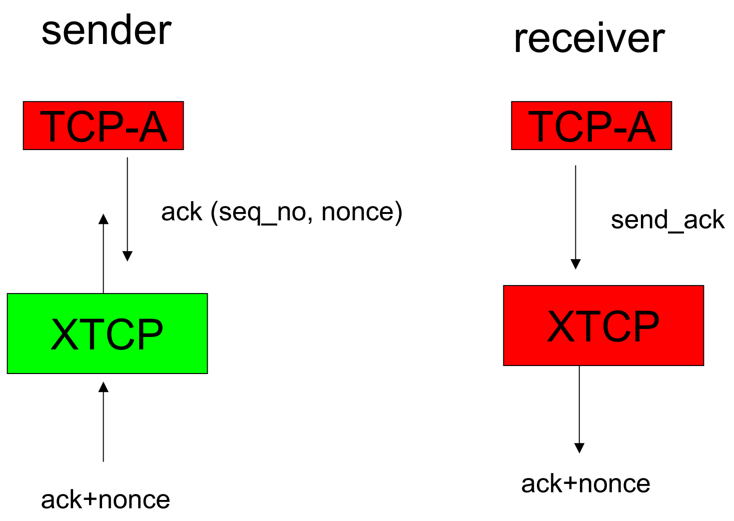
datagram with nonce

🟥 TCP-A

🟥 XTCP

Datagram with nonce

Nonce Mechanism



sender

receiver

🟥 TCP-A

ack (seq_no, nonce)

🟩 XTCP

ack+nonce

🟥 TCP-A

send_ack

🟥 XTCP

ack+nonce

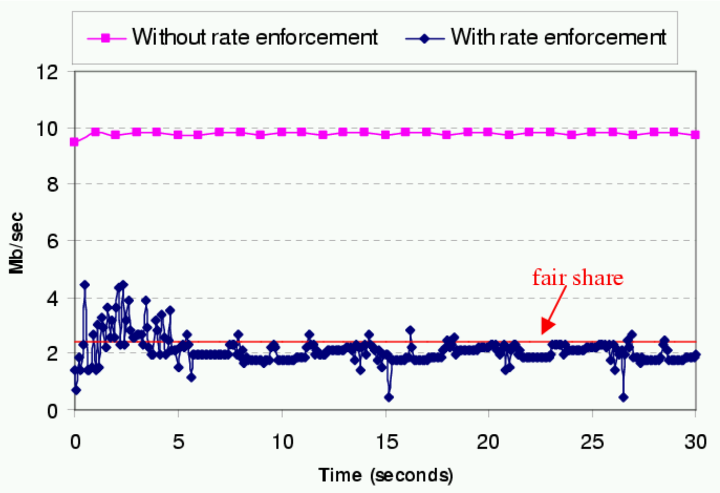
3. Performance

- Connections proceed without delays
  - Code is downloaded out-of-band
  - Benefits later connections
- Efficient to share data between the C-based kernel and Cyclone code
  - No garbage collection
  - Lightweight runtime

Status

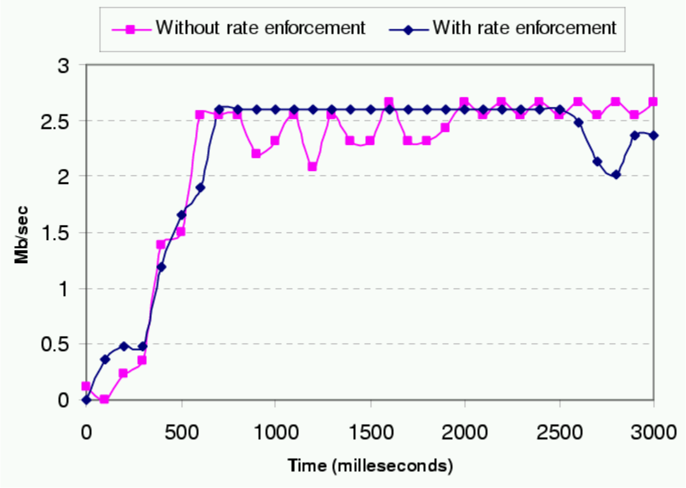
- Prototype in FreeBSD 4.7
- Modest memory and CPU cost
  - CPU cost is 80% more than base TCP on the sender side, without any optimizations.
- Ported TCP Friendly UDP, TCP NewReno and TCP SACK to the XTCP API
- User-level version in progress

It works! UDP



fair share

It works! TCP



Open research issues

- TCP rate policing function
  - Quickly detect unresponsive extensions
  - Admit all responsive extensions
- XTCP API
  - Must be sufficient and portable

Conclusions

- Transport protocols need self upgrade mechanism
- Mobile code works !
  - Constrained domain and recent advances
    - Mathematical definition of TCP response function (1998)
    - Cyclone (2002)

END OF TALK

….

BACKUP/DETAIL SLIDES

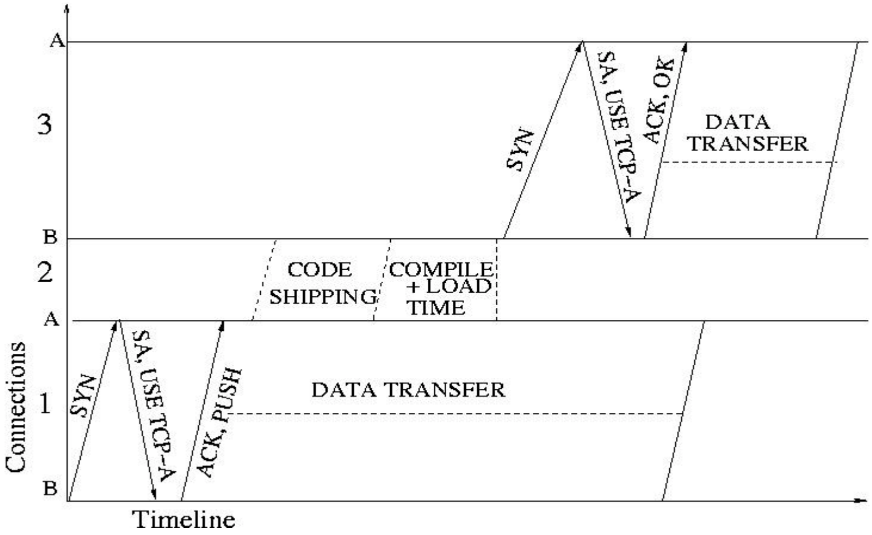
Policies

- Applications can use socket options
- Administrators can set policies
- Policy daemons can collaborate

Granularity of extensions

- Transport protocols are written to the XTCP API
- Complete transport protocols are transferred over the network
  - Retains the way protocols are written now
  - Maximum flexibility
  - Maximum simplicity
  - Code is not large: 85K compressed source

Connection setup and code loading



Network safety

- TCP friendly network access API
  - ◆*xtcp_net_send* (seq_no)
  - ◆*xtcp_net_resend* (seq_no)
  - ◆*xtcp_net_ack* (seq_no, nonce)
  - ◆*xtcp_net_acksum* (seq_no, nonce_sum)

A Fourth Challenge:
Deployment of XTCP framework

- Benefits self
- Can only harm self
- Deployment only needed at end points
- TCP-friendliness is non-threatening