Transport Futures

IETF 66 - Montreal
TSV Area Meeting

Mark Allman
mallman@icir.org

"Would it satisfy ya?
Or, would it slide on by ya?
Or, would you think the boy is strange?"

Background

- TCP has clearly served us well

- Seeing things crop up now that represent TCP *protocol* changes
  - ▶ as opposed to algorithmic / behavioral changes

- Changing TCP's protocol is getting increasingly painful
  - ▶ too few bits

- We still use TCP for a few things…so, what should we do about twisting TCP to accommodate protocol changes?

Background (cont.)

- Advertised window space is too small
  - ▲ long "solved" by window scaling (RFC 1323)
  - ▲ led to increased likelihood of wrapped sequence space
    - ■ hence, timestamps for PAWS

Background (cont.)

- Option space is anemic
  - ▲ could be growing more scarce
  - ▲ IETF work items chewing space: UTO, Quick-Start
  - ▲ others possible: TCP-Auth, portnames, etc.
  - ▲ two concrete proposals for extending the option space
    - ■ Long Options Option (Eddy)
    - ■ Extended Option Space (Kohler)

Background (cont.)

- Port numbers
  - ▲ as noted on the IETF list, some very busy servers are taxing the port space
  - ▲ two concrete schemes that would aid port space usage
    - ■ port migration (Shepard)
    - ■ portnames (Touch)

Background (cont.)

- Reserved bits
  - ▲ 3 bits remain
  - ▲ will we ever feel compelled to burn any more?

Goal

- Given that we're seeing solutions to these "problems" crop up, let's consider the space

- Make an explicit decision, rather than implicitly following some path or pocket vetoing another avenue

Approach #0

- Do nothing

- Decide that these problems are not really all that bad and just decide that we can live with the current header field limitations

"Whisper words of wisdom, let it be"

Approach #1

- Do the best engineering we can on each individual problem as the problem appears

"I saw her today at the reception, a glass of wine in her hand"

Approach #2

- We have defined two "modern" transport protocols in DCCP & SCTP

- So, we could end-of-life TCP from the perspective of enhancing the protocol

  ▶ possibly modulo acute security issues

"Well, I started out...down a dirty road"

Approach #3

- We could decide that...
  - ▶ DCCP & SCTP don't cut it
  - ▶ we want a reliable, in-order byte-stream transport in our toolbox
  - ▶ and, current TCP is not that tool

- So ... open the gates on a TCP-ng

"Out in the great wide open, a rebel without a clue"

Approach #4

- We could try to put a new header on an old protocol

- Crazy idea: TCPx2
  - ▲ double the size of the TCP header fields
  - ▲ grab a new IP protocol number
  - ▲ retain the protocol semantics
  - ▲ draft-allman-tcpx2-hack-00.txt

"You ain't a beauty, but eh you're alright"

Approach #4 (cont.)

- Provides applications fundamentally the same service
  - ▶ reliable, in-order byte-stream

- Keep semantics, just provide for larger values
  - ▶ leverage large portions of existing code
  - ▶ and, understanding

- There is an easy fallback to current TCP
  - ▶ send a TCPx2 SYN, retransmits use standard TCP

"You ain't a beauty, but eh you're alright"

Approach #4 (cont.)

- There are plenty of issues
  - ▲ implementations need changed
  - ▲ middleboxes will get confused (!)
  - ▲ etc.

- It is not trivial nor a thing of beauty, but is it reasonable?
  - ▲ one hack instead of N hacks

"You ain't a beauty, but eh you're alright"

So ...

- What do you think about the *space*?

"Sometimes you're the windshield, sometimes you're the bug"

Approaches

0. Do nothing
1. Handle each field as problems arise
2. End-of-life TCP in favor of more modern transports
3. Blank slate reliable, in-order, byte-stream protocol
4. Mark's crazy hack

"Sometimes you're the windshield, sometimes you're the bug"