

Quo Vadis, DCCP?

Pasi Sarolahti & Tom Phelan

IETF-75, Stockholm

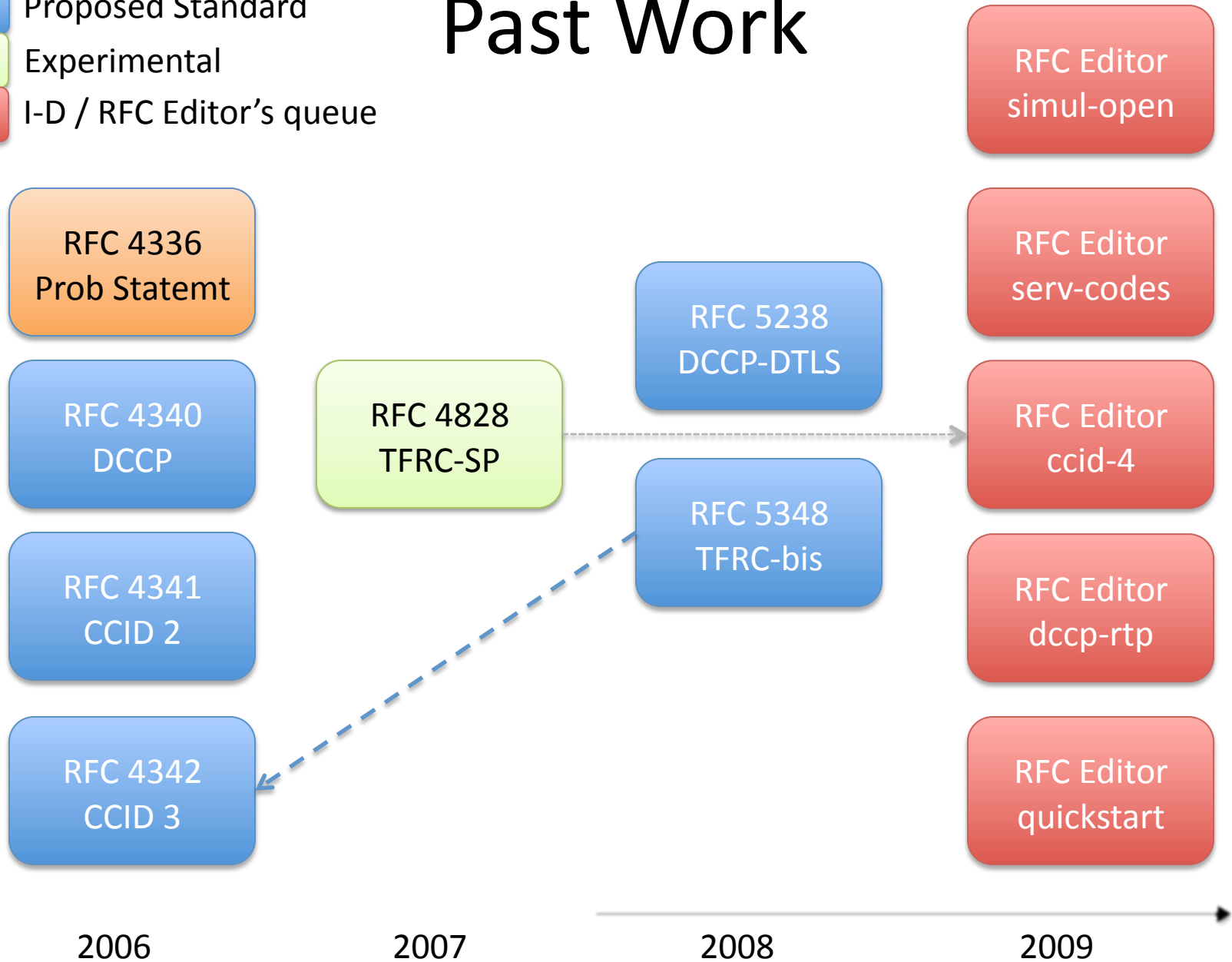
July 27, 2009

DCCP in a Nutshell

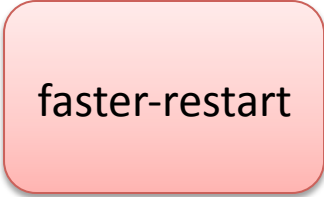
- Standard solution for **reliably transmitting congestion feedback** for unreliable datagram flow
 - End-to-end transport protocol
 - WG started in 2002
- Choice of congestion control mechanisms
 - TCP-like congestion control
 - TCP-friendly rate control (TFRC)
 - TFRC for small packets
- Other features
 - ECN support, partial checksums, etc.

- Informational
- Proposed Standard
- Experimental
- I-D / RFC Editor's queue

Past Work



Currently Open WG Items



faster-restart

(Expired)

Implementations

- Linux kernel
 - CCID-2
 - CCID-3
 - NAT implementation
 - CCID-4 and ECN support in progress
- DCCP-TP
 - User-space implementation optimized for portability
 - CCID-2
 - CCID-3 with RFC 5348 (RFC3448bis)
 - DCCP-NAT encapsulation (draft-phelan-dccp-natencap)
 - Fresh start code – no code shared with Linux kernel implementation
 - See <http://www.phelan-4.com/dccp-tp/>

Main Current Challenges

- **Middleboxes** don't handle DCCP packets
 - Significant disincentive for turning on DCCP
- Better **APIs** to communicate congestion/rate information would improve efficiency
- Not much experience on **congestion control algorithms** with real applications
 - Potentially room for improvement

Ideas for New Work on DCCP

- UDP framing for X
 - draft-phelan-dccp-natencap DCCP specific, is there general solution?
- DCCP as generic congestion control framework
 - Use congestion feedback channel for new types of applications
 - Support innovative uses of explicit congestion signals
 - Feedback for adjusting (en)coding algorithms
 - New types of distributed content sharing, games
- Better congestion control algorithms
 - Beyond TCP-*??

MuTFRC

draft-welzl-multfrc-00.txt, <http://www.welzl.at/research/projects/multfrc/index.html>

- CC. mechanism which is “N-TCP-friendly”
 - N can also be 0.3, 2.8, ...
- More appropriate behavior than multiple real TFRCs
 - See talk in ICCRG meeting for more details
- Proposal: specify mechanism (like TFRC), then CCID
 - Implementation: a handful of simple changes to TFRC code
- Better bottleneck saturation while still being reasonably TCP-friendly
 - N limited to 6 in draft; yields 95% utilization of otherwise empty bottleneck; only 75% with 1 TCP-friendly flow
 - Could this create an incentive to use DCCP?

WHERE TO GO, DCCP?