Linux *(TM)* DCCP Implementation Feedback

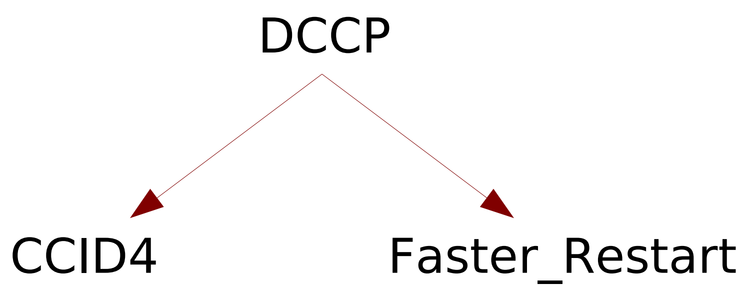
G. Renker, Ian McDonald,
Arnaldo Carvalho de Melo,
T. Saviranta, L. M. de Sales,
A. Bittau

Current Development

- DCCP/CCID3 – one maintainer, one developer
- CCID4 – two developers
- Faster Restart – one developer

Test Tree

- *mainline wants production-ready code*
- *but DCCP still has many experimental aspects*
- *purgatory* for patches (currently merging)



`git://eden-feed.erg.abdn.ac.uk/dccp_exp`

Kernel Maintainer Feedback

- Arnaldo Carvalho De Melo
- making DCCP a *first-class network stack citizen*
  - as part of a mainstream OS
  - efficient integration with existing protocol stack
  - improved maintainability
- *steady and continuous progress* in code revision
- *input is solicited* how DCCP is being
  - used (how, where, settings, apps, ...) ???
  - tested (results, comparisons, ...) ???

General Feedback

- changing format of rfc3448bis hinders progress
  - *interdependencies* cause problems
  - 4 developers refer to 3 different draft versions
- RFC1323-algorithm needed for RTT estimation
  - *principle* is simple (Timestamp + Elapsed Time)
  - but *details* are complicated & non-trivial
    - deal with duplicate timestamps, reordering, delay
    - RFC1323 didn't get it right in the first place
    - cf. draft-ietf-tcplw-high-performance-00
  - would help *much* to improve internals

CCID3 Feedback

- problems with receiver-RTT estimation
  - X_recv accuracy depends on RTT accuracy
  - algorithm gets confused by the min CCVal = 5
  - RTTs are influenced by packet-timing compression
    - EWMA filter helps, but RTTs appear much higher
    - very messy to filter out marginal conditions
- suggestion: sender communicates his/her RTT
  - sender has a very accurate RTT estimate
  - originally suggested in RFC 3448
  - could use a DCCP option?

CCID4 Feedback

- should reported X_recv be used *as-is*?
  - should application run the values through *"smoothing" function* before using new value?
  - e.g. using a standard EWMA filter?
- calculation of average loss interval in TFRC-SP:
  - the most recent loss interval is used in calculation only if it's "*long*" (e.g. >= 2 RTT)
  - is this *sufficient* for senders not validating X_recv against reported loss intervals and dropped packets?

CCID4 Feedback: options

- Loss Intervals / Dropped Packets: fields too big?
  - for Lossless Length, Loss Length, and Data Length
- lossy part of Loss Interval cannot be > RTT:
  - 24-bit counters appear to be over-dimensioned
  - especially with CCID 4 (sends at most 100pps)
- due to feedback once per RTT [RFC4828]:
  - Lossless Length and Data Length fields might also be shorter
  - 16 bit or even less?

Faster Restart Feedback

- implementing X_recv_set seemed too complicated
  - so implementation just used X_recv
  - i.e. as per rfc3448bis-00
- in present tests Faster Restart showed *no noticeable improvement*
    - but may be due to selection of test scenario
    - contact Ian McDonald for further information

Growing list of DCCP applications

- *VLC* (video/audio streaming)
  www.videolan.org/vlc
- *paraslash* (audio streaming)
  paraslash.systemlinux.org
- *gstreamer plugin* (VoIP, streaming)
  gstreamer.freedesktop.org
- *SpeexComm* (VoIP application)
  tuomas.kulve.fi/speexcomm