DCCP changes, open issues, & implementations

—

Eddie Kohler, Mark Handley, Sally Floyd, Jitendra Padhye
DCCP Working Group
July 15, 2002—Yokohama IETF

http://www.icir.org/kohler/dccp/
Review

- UDP plus congestion control plus reliable feature negotiation
  - Unicast
  - Bidirectional data transfer
  - Selectable congestion control mechanisms
  - Per-packet sequence numbers
Changes since SLC IETF
Changes since SLC IETF
Changes since SLC IETF

- Changed name to Datagram Congestion Control Protocol
  Acronym sounds less like TCP
Other clarifications

- Changed feature negotiation options
  
  Ask $\rightarrow$ Change
  Choose $\rightarrow$ Prefer
  Answer $\rightarrow$ Confirm

- Expanded acknowledgements discussion
  
  Unidirectional communication/quiescence

- Checksum includes a pseudoheader
Illustrative

Restricted to app events, timers, receiving valid packets

Packets without explicit transitions are invalid
Sequence number validity

- What sequence numbers are valid?
  For instance, when to ignore a Reset?

- Partial solution: Loss Window feature
  Like maximum number of packets sender expects to be in flight
  Defaults to 1000

- Problem: sequence numbers change with every packet, even Acks
  Can get out of sync relative to any window

- Solution: Connection Proof
Connection Proof

- Each endpoint has a Connection Nonce
  Short random string
  Trade nonces during connection setup
- Connection Proof option: xor of nonces
  Proves you know both nonces
- Resync with Identify Yourself option, which requests Proof
  Receive invalid seqno → Ack with Identify Yourself
  Response has valid Proof → resync to that seqno
- Needs more thought (security?)
Mobility

- New Move packet format supports IP6

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Reserved | Acknowledgement Number |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Old Address Family | Old Port |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Old Address |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| [padding] |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Options | [padding] |
```

- Also, Move uses Connection Proof
Receiver alerts

- Already had Receive Buffer Drops
  Packet in receiver kernel due to kernel space

- New Slow Receiver option
  Packet not dropped, but receiver having trouble keeping up
  Running low on buffer space, CPU time, quotas ...
  Sender responds by not increasing sending rate
  Better than receive window

- New Buffer Closed Drops option
  Application has closed receiving socket
CCIDs

- Removed CCID 0 “Single-Window Congestion Control”
  Intended for endpoints that want to hold minimal state
  But you can hold minimal state without CCID 0
  More trouble than it was worth

- CCID 3 clarifications and corrections
  “Design Considerations” section
Open issues
Open issues

C
Open issues

- DCCP = “Distributed Checksum Clearinghouse Protocol”? 
  Ethereal thinks so
  The protocol designers don’t
  Crap
API

- On Unix, DCCP will probably use a socket interface
  Connection establishment and teardown

- Kernel communication
  Optional minimal kernel buffering? (Delay sending packet until CCID approves)
  Set CCIDs
  Slow Receiver, Buffer Closed Drops upcalls?
  Share sequence numbers with user level?

- What level of specification is appropriate, and in which draft?
RTP over DCCP

- Problem: duplicate sequence numbers and receiver reports

- Solution 1: There is no problem, layer as is
  - RTP applications use seqnos differently (ordering, ARQ, ...)
  - Receiver reports: DCCP interested in CC, RTP in application
  - Extra space cost not overwhelming
  - Premature optimization, blah blah blah

- Solution 2: Develop optimized RTP header for layering over DCCP
  - Elide sequence number, receiver reports when possible
Requested extensions

- Bright line: “Only if you can’t layer it above”
  But, for the sake of discussion . . .
- Multiplexing (subflows/streams)
  Option: “This packet is part of subflow K”
- Fragmentation
  Currently prevented from sending datagrams larger than MTU
  Options: “First fragment”, “middle fragment”, “last fragment”
  Only deliver to app when reassembled; no automatic retransmission
- Selective reliability (API changes only?)
Others

- Quiescence
  
  Anecdotal evidence: difficult to implement
  
  It is only an optimization

- Connection Proof, receiver alerts, security
  
  More thought and/or discussion

- Receiver window
  
  Is Slow Receiver sufficient?
Implementations

- Patrick McManus: Linux kernel
  Pretty full-featured
  http://www.ducksong.com:81/dccp/

- Berkeley [Sohn, Zolfaghari, Evlogimenos, Lim, Lai]: user level
  Simplified; for instance, only CCID 3
  http://www.cs.berkeley.edu/~laik/dcp/

- Neither implements quiescence (I think)