Designing DCCP: Congestion Control Without Reliability

Eddie Kohler, Mark Handley, Sally Floyd
SIGCOMM '06

Steffie Jacob Eravuchira
2014
Outline

- Introduction
- Application Requirement
- DCCP Overview
- Sequence Numbers
- Connection Management
- Congestion Control
- Conclusion
Introduction

- Who prefers timeliness over unreliability?
  - Streaming media
  - Internet telephony
  - Video conferencing
  - Games

- Why not TCP / UDP?
Application requirement

- Goals
  - Minimal functionality and mechanism
  - Robustness to attacks and middleboxes
  - Framework for congestion control
  - Self-sufficiency
- Timing – reliability tradeoff
Application requirement...

- Deliberate omissions
  - Flow control
  - Selective reliability
  - Streams
  - Multicast
DCCP Overview

- What is DCCP?
  - Unicast
  - Connection-oriented
  - Bidirectional
DCCP Overview...

- Three-way handshake during initiation and termination
DCCP Overview...

- DCCP Header
Sequence Numbers

- Determines DCCP's connection management features

- TCP
  - 32-bit seqno
  - Cumulative ackno or SACK
  - Corresponds to bytes of data
  - No seqno for pure acknowledgement
Sequence Numbers...

- DCCP
  - Header of all packet types contains \texttt{seqno}
  - Measure datagrams, not bytes
  - Most packet has:
    - \texttt{Seqno}
    - \texttt{Ackno} to latest packet received
    - Additional \textit{Ack Vector} (64 bit)
Sequence Numbers...

- Synchronization
  - Expected-sequence-number window
  - Large burst of loss leads to loss of sync
  - Explicit synchronization
    - Sync
    - SyncAck
Explicit synchronization

(a) Synchronization

Data(seq 0) → Expect seqnos [0, 100]

Data(seq 1) → Odd seq, send Sync

... →

Data(seq 500) →

Data(seq 501) →

Sync(seq 1, ack 501) →

SyncAck(seq 502, ack 1) → Update to [502, 602]
Half-open connection recovery problem
Half-open connection recovery solution
Sequence Numbers...

- Acknowledgements
  - TCP - Bounded state, *cumulative ackno*

- DCCP
  - Unbounded state, *Ack Vector*
  - *Ack-of-ack* required to prune receiver state
  - Ack does not guarantee that packet is delivered to application (to tradeoff timeliness to reliability receiver may drop packets)

- Data Dropped option
Sequence Numbers...

- Sequence number length
  - Short seqno has wrapping problem
- 48-bit solves wrap problem but creates header overhead
- Solution – leave off upper 24-bit value
Sequence Numbers...

- DCCP header with short sequence number
Sequence Numbers...

- Receiver takes 24-bit value $s$ and an expected sequence number $r$ and returns $s$’s 48-bit extension

\[
\begin{align*}
    r_{\text{low}} & := r \mod 2^{24}; \quad r_{\text{high}} := \lfloor r/2^{24} \rfloor; \\
    \text{if } (r_{\text{low}} \oplus s < r_{\text{low}}) & \quad \text{// $s$ incremented past } 2^{24} - 1 \\
    \quad \text{return } ((r_{\text{high}} + 1) \mod 2^{24}) \times 2^{24} + s; \\
    \text{else if } (s \ominus r_{\text{low}} < s) & \quad \text{// $s$ decremented past 0 (reordering)} \\
    \quad \text{return } ((r_{\text{high}} + 2^{24} - 1) \mod 2^{24}) \times 2^{24} + s; \\
    \text{else} & \\
    \quad \text{return } r_{\text{high}} \times 2^{24} + s;
\end{align*}
\]
Sequence Numbers...

- 48 bit seqno
  - Connection initiation, synchronization, teardown packets
  - Endpoints agree on full value
  - Reduce attack possibility

- 24-bit seqno
  - Data and acknowledgement packets
  - Major portion of connection with less overhead
Sequence Numbers...

- Robustness against attacks

1. TCP
   - Secured sequence number
   - Cryptography prevents snooping attack
   - Data injection requires guessing $\text{seqno}$ and $\text{ackno}$

- Connection reset attack easier using SYN
Sequence Numbers...

2. DCCP

- Reset attack is harder because of 48-bit seqno and explicit synchronization

- Data injection is easy if data packets with 24-bit seqno and no ackno. But less dangerous

- Data injection may cause connection reset using 'options'

- Options in data packet must be ignored
Connection Management

- Asymmetric communication
  - Single bidirectional connection
- Spend little on inactive space
- A half-connection?
- Bidirectional communication has both halves active and acknowledgements piggybacked on data packets
Connection Management...

- Two half-connections and a full-connection with piggybacked data and acknowledgement
Connection Management...

- Quiescent half-connection (One which is inactive for 0.2 seconds or 2 RTTs, whichever is greater) need not be acknowledged except to prune its state

- Independent congestion control method in each half-connection
Connection Management...

- Feature Negotiation
  - Endpoints must agree on choice of congestion control methods, seqno length etc in band

- Two option types here:
  - Change options
  - Confirm option
Connection Management...

- Denial of Service attack
- TCP – SYN attack

- DCCP
  - push state to client whenever possible (Init Cookie)
  - Time-Wait state on willing client
  - CloseReq by server
Connection Management...
Congestion Control

- Congestion Control IDs (CCIDs)

- CCID 2: TCP-like Congestion Control
  - Ack Vector
  - Congestion window
  - Unlike TCP, reverse-path maintains Ack Ratio
Congestion Control...

- CCID 3: TFRC Congestion Control
  - Sending rate
  - Receiver reports a set of loss intervals
  - Loss interval contains maximum tail of non-dropped, non-marked packets
  - Sender calculates a loss event rate
  - $CCVal$ (4bytes) - Data packets need to attach timestamp to data packet, every quarter of the RTT to group losses
Congestion Control...

- Future Congestion Control issues
  - Sending many small packets rather than fewer large ones.
- Rapid startup after idle periods
- Abrupt changes in application data rate
Congestion Control...

- Misbehaving receivers
  - Pretending lost packets were received, or that ECN-marked packets were received unmarked
  - Protocol must detect deliberate behavior

- Solution:
  - Sender provide unpredictable nonce
  - Receiver echoes an accumulation of all relevant nonces
Congestion Control...

- Partial checksums and non-congestion loss
  - Applications prefer corruption to loss
- UDP-Lite
- CsCov field – restricts checksum coverage
- Corruption and loss reported separately
Conclusion

- Explicit synchronization
- New acknowledgement format
- Modular congestion control mechanism
- Robustness against attack
Reference