WebRTC Data Channels

draft-jesup-rtcweb-data-02

Randell Jesup
Salvatore Loreto
Michael Tuexen

IETF Interim
Updated from IETF 82 presentation

Uses

- Side channels during a 'call' (mute status, etc)
- Chat
- File transfer
- Application synchronization
- Games
- Shared whiteboard
- Co-browsing
- Shared document editing (with audio and/or video)
- Many uses we haven't thought of yet

Data Channel Requirements

- Multiple data channels
- Reliable and unreliable
- Datagram and Stream (if reliable) paradigms
- MUST be congestion-controlled
- MUST be secure (*)
- Quality open-source userland implementation needed for deployment
- See draft for other implementation requirements

Options

- Pseudo-TCP-over-UDP (reliable) + DCCP (unreliable), both over DTLS-(ICE)-UDP
  - Pseudo-TCP: no specification; in-use with source code
  - DCCP: specification; no user land implementation

- SCTP-DTLS-(ICE)-UDP or
- DTLS-SCTP-(ICE)-UDP
  - DTLS-SCTP specified (RFC 6083), SCTP-DTLS not currently (believed to be straightforward)
  - Provides reliable, unreliable, partial-reliable, datagrams and streams

Pseudo-TCP-over-UDP (reliable) + DCCP (unreliable)

- Pros
  - Well-known protocols
  - Open-source pseudo-TCP available
- Cons
  - Two protocols needed
  - Loss-based congestion control (DCCP CCID3 is similar to TFRC)
  - No known-stable user-space DCCP available
  - Multiple congestion-control flows (fights between flows)

SCTP-DTLS-(ICE)-UDP or DTLS-SCTP-(ICE)-UDP

- Pros
  - Single kitchen-sink protocol
  - Open-source userspace implementation based on FreeBSD
  - Direct support for stream API (in SCTP-DTLS)
  - Option of partial-reliability and out-of-order delivery
  - Single congestion-control flow
- Cons
  - Limitations sending large datagrams (but SCTP-DTLS can use streams)
  - Loss-based congestion control (but replaceable)
  - SCTP-DTLS has no draft currently (shouldn't be a problem)
  - Single receive window (see Open Issues)

SCTP-DTLS-(ICE)-UDP vs DTLS-SCTP-(ICE)-UDP

- SCTP-DTLS
  - Direct use of the SCTP API
    - Such as reliable-channel streaming, partial-reliability, etc
  - No draft, though should be straightforward
  - Interleaving of large datagrams can (easily) be added to SCTP
- DTLS-SCTP
  - Can use kernel implementation (browsers generally won't, though)
  - DTLS-SCTP specified in RFC 6083.
  - Reliable channels would be datagrams, not streams (or needs an extra layer)

Open issues

- SCTP
  - Michael Thornburg's issues
    - Blocking of other channels if one isn't serviced
  - Draft for SCTP-DTLS needed if chosen
  - Interleaving of large datagrams
- DCCP
  - Is a userland implementation available? Quality?
- General
  - Inter-stream priority (nice-to-have)
  - Congestion control interactions with app and media streams
  - PMTU sensing

Progress since IETF 82

- Updated userland SCTP released (Win/Mac/Linux)
- API work by Justin Uberti

Congestion Control

- SCTP supports pluggable congestion control
- We want to have the data channels coexist with the delay-sensitive congestion control planned for the media streams
  - Some type of priority algorithm – must be fair, but must be weightable
  - Avoid starving media channels when doing large data transfers
  - Minimize delay sending data in sparse data channels
  - Must work when competing with large TCP flows and not
  - Ideas:
    - Bandwidth set as % by with optional min/max caps
    - Cx-TCP
    - Default TCP-like or optional TFRC-compatible modes

Bandwidth % and caps

- The bandwidth allocated to the data channels could be expressed as a % of total the media channel believes is available
- Optional top and bottom caps would be a good idea
- % set a a result of channel priorities
- To use those bits for media when not used by data, would need to allow the media channels to use bits (very) recently not used by the data channels.
  - Perhaps in period N let media encoders use unused data bits from period N-1 – period must be short << 1s
- Implies data is fed to some type of output queue scheduler
- What do we do when there's still loss?

Cx-TCP

- Possible solution: replace congestion module with one based on Cx-TCP (Budzisz, Stanojevic, Schlote, Baker et al)
  - Cx-TCP is a delay-sensitive TCP congestion algorithm shown to be fair with TCP flows and other Cx-TCP flows
  - Cx-TCP approximates RED AQM; typically keeps delays low (~20ms in their recent paper)
  - Open investigation would be to prove fairness with algorithms based on methods derived from Harald's draft
  - Further investigation required to ensure this is usable in low-load situations as it was designed for high-utilization links

TCP and TFRC-like control

- We could always use the default TCP-like or TFRC-like congestion control algorithms
  - Violates requirement to avoid starving media channels; would likely need some way to limit maximum BW use
  - Easy

Questions/Discussion

- Is there consensus on using SCTP? (I think yes)
- If so, what are people's opinions on ordering with DTLS?
  - What information is needed before consensus can be reached?
- What congestion control method should be used?
- What does the API for different Data Channel options look like? (W3C)
- What does the API for opening Data Channel channels look like? (W3C)