

Unidirectional Streams in Minq

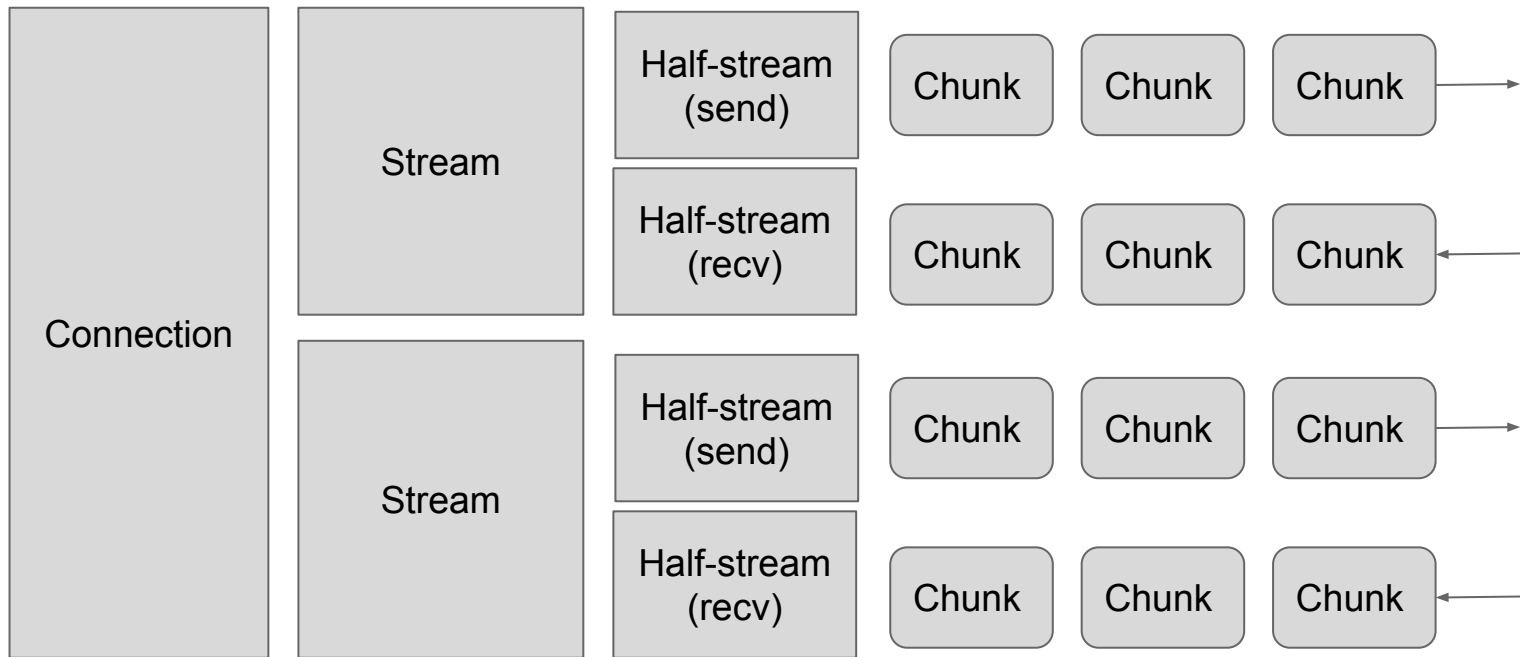


Eric Rescorla
ekr@rtfm.com

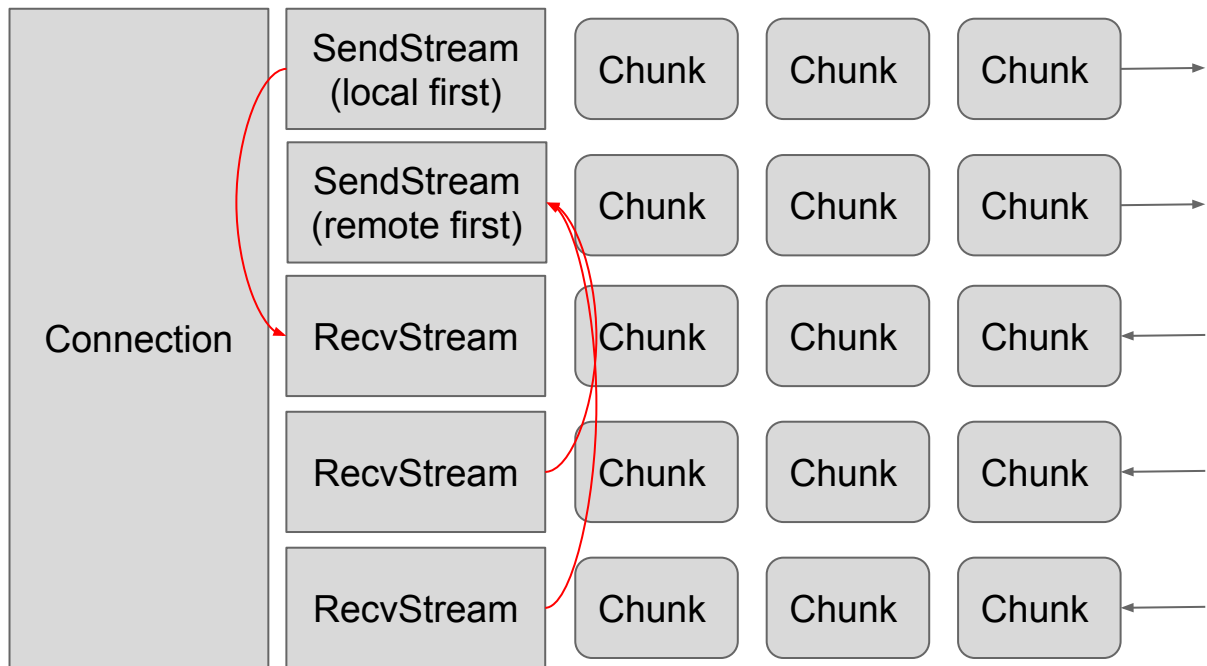
Overall Status

- `Minq master` currently implements -05
 - With HTTP/0.9
- `Minq unidirectional_streams` branch implements (QUIC-only)
 - PR #643: "Unidirectional Streams"
 - PR #720: "Add bidirectional streams on top of unidirectional"
 - A bidirectional "unified" stream API on top (nearly the same API as `master`)
- Total time investment to adapt to unidirectional: ~16 hours

Architecture for -05



Architecture for Unidirectional Streams



Bidirectional Streams API

- `Connection2` is a bidirectional wrapper for `Connection`
 - Actually, `Connection` is a mixin
- `Stream` is a pair of `SendStream` and `RecvStream`
 - API calls mostly go to the underlying directional stream
- Still working out `Close()`
 - But that's because we don't understand semantics
- Possible to use bidirectional streams API with a “conformant” related-streams peer (my test programs work this way)
- Note: this won't work with many-to-one related mappings

Bidirectional Streams Internals

- Streams locally created with `CreateStream()`
- Remote streams notified with `NewStream()` event
- When locally created (send first), starts with an empty `RecvStream`
 - Writes at this point appear to block
 - `RecvStream` automatically filled in when a related recv stream appears
- When remotely created (recv first), we secretly create a paired `SendStream`, ready for use

Impact on Applications

- **Straightforward API call mapping**
 - `GetReceiveStream()`, `CreateSendStream()`, `CreateRelatedSendStream()`
 - Bidirectional protocols need a bit of work
 - With remote-first streams, do `CreateRelatedSendStream()`
 - With local-first streams, `Connection` calls `NewRecvStream()` callback
- **With bidirectional API, mostly just search and replace**
 - `s/Connection/Connection2/`

Disadvantages of unidirectional streams

- A bit more work for bidirectional protocols
 - But bidirectional API hides this
- Semantics of closure are kind of unclear
 - What API should we provide? (`close()`, `shutdown()`)
 - What API should I use if I don't like a remotely created stream
- Easier for sides to disagree about mapping
 - Is this stream unpaired, 1:1, or 1:N?
 - It's not signalled inband right now
 - This will need to be specified in the protocol
- "Related streams" header inclusion rules are a bit awkward
 - Proposal: require in all stream packets till one ACKed

Advantages of unidirectional streams

- Was easier and more natural to implement
 - "Stream halves" don't really make sense
 - Composition let me share the common pieces
 - Simpler state machine (e.g., `Reset()` always goes to `CLOSED`, not sometimes to `HC-Local`)
 - No goofy odd/even semantics
- Clearer semantics around remote creation
 - In bidirectional, if I receive `STREAM_MAX_DATA` can I send?
- More flexible semantics
 - unpaired, 1:1, or 1:N mappings