The future web: HTTP 3 in Rust

Dirkjan Ochtman
Rust Nijmegen, March 2019

https://github.com/djc/quinn

About me

 **@djc**  **@djco**

- https://dirkjan.ochtman.nl/

- Rust projects: Askama, tokio-imap

- Software engineer on ING's Distributed Ledger Technology ("blockchain") team

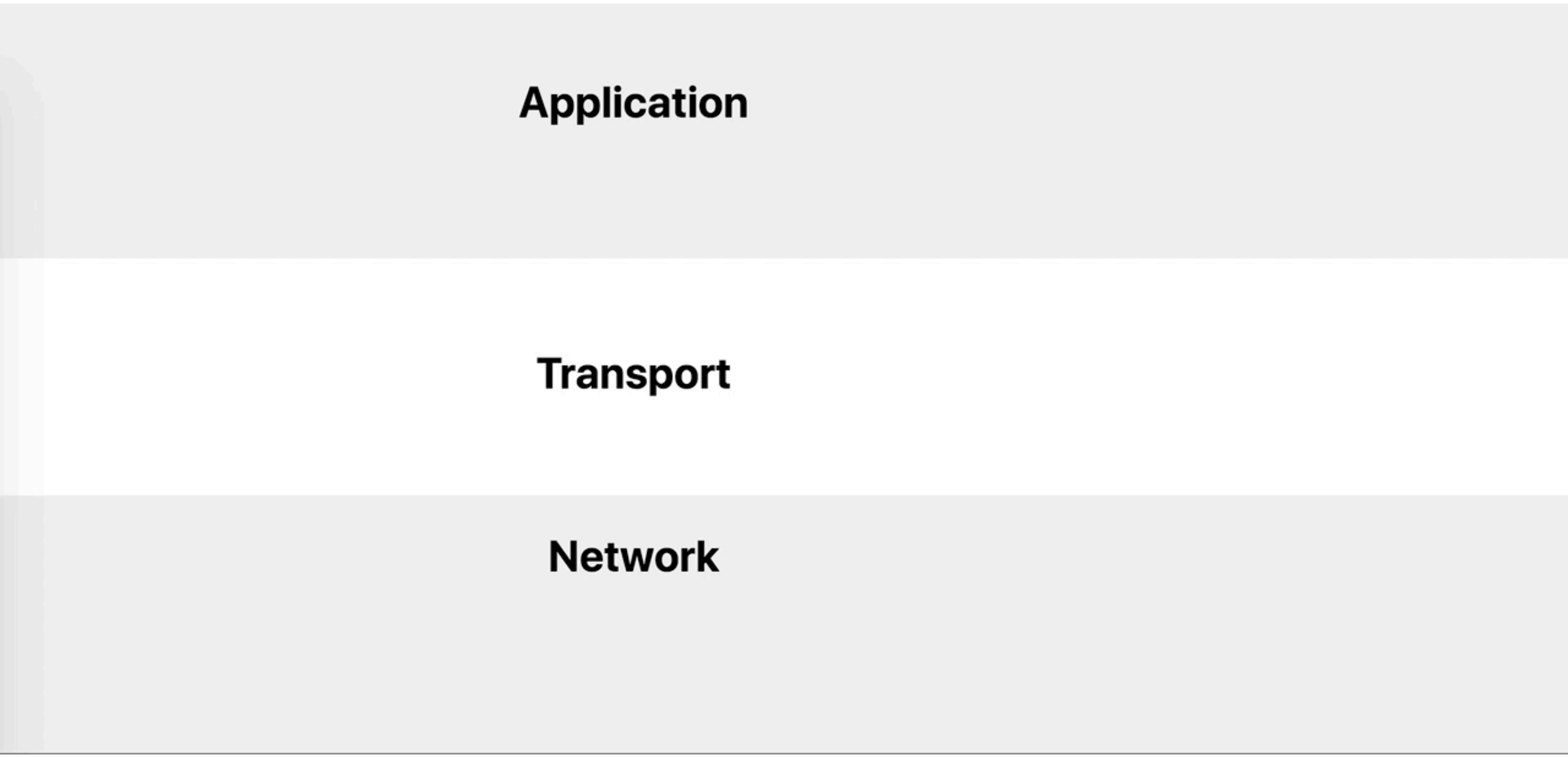
What is HTTP 3?

- Reformulating HTTP (2) on top of the QUIC transport
- Re-envision TCP after ~40 years of use
- QUIC: started at Google in 2012, IETF working group established in 2016, planning for final drafts in Nov 2018
- (SPDY announced 2009; first draft of HTTP 2 in 2012; RFC 7540 published May 2015; SPDY removed in 2016.)
- Expect QUIC v1 and HTTP 3 RFC in Q3 2019

Table of contents

- **QUIC: problem statement, concepts and process**
- Quinn: design, implementation status, future

Head-of-Line blocking



QUIC goals

- Minimize latency
- Provide multiplexing
- Changes limited to path endpoints
- Support connection migration
- Provide always-secure transport
- Expect rapid distributed development and testing

QUIC documents

- Invariants: things that cannot change in future versions
- Transport: core protocol logic
- Recovery: loss detection and congestion control
- TLS: how to secure the transport payloads
- QPACK: header compression (static & dynamic)
- HTTP: mapping of HTTP (2) semantics to QUIC

Recovery

- Differences with TCP
  - Monotonically increasing packet numbers
  - No reneging (reduces memory pressure on sender)
  - More ACK ranges (less reliant on timeouts)
  - Explicit correction for delayed acknowledgements

QUIC TLS

- TLS 1.3 handshake messages with a custom extension
  - Extension for announcing transport parameters
- After handshake, encrypt payloads with negotiated secret
  - Application data is not wrapped in TLS records

HTTP 3

- ALPN protocol "h3" negotiated during TLS handshake
- Builds on HTTP 2, differs in order to leverage QUIC
    - Different header compression scheme
    - Different framing mechanism

Interoperability matrix

client ↓	h2o/quicly	quant	ngtcp2	mvfst	picoQUIC	winquic	f5	ATS	quiche	lsquic	ngx_quic	AppleQUIC	quic-go	Quinn	P
h2o/quicly															
quant		VHDCRZS MBUPE	**VHDCRZS MBU**		**VHDCRZS MBU3P**	**VHDCRZS P**	VHDCRS	**VHDCRZS M3**	VHDCRS	**VHDCRZS M3E**	VHDCR			**VHDCRZS MBUPE**	
ngtcp2		**VHDCRZS U**	VHDCRZS MBU		**VHDCRZS MBU**	**VHDCRZS**	**VHDCRS B**	**VHDCRZS MB**	VHDCRS	VHDCS				**VHDCRZS MBU**	
mvfst															
picoQUIC		**VHDCRZS MBUP**	**VHDCRZS MBU**		VHDCRZS MBU3P	**VHDCRZS P**	**VHDCRS B**	**VHDCRZS MB3**	VHDCRS	**VHDCRZS MB3**	VHDCR	VHDC		**VHDCRS MBUP**	
winquic		**VHDCRZS**	**VHDCRZS**		**VHDCRZS**	VHDCRZS P	VHDCS	**VHDCRZS**	VHDCRS	VHS				VHDCS	
f5		HDCS	HDCS		HDCS	HDCS	HDCS	HDCS	HDCS	HCS				HDCS	
ATS		**VHDCRS M**	**VHDCRS M**		**VHDCRS M**	VHDCRS	VHDCRS	VHDCRS M	VHDCRS	VHS		VHDC		**VHDCRS M**	
quiche															
lsquic					**VHDCRS 3**			**VHDCRS 3**		VHDCRS 3					
ngx_quic	-	-	-	-	-	-	-	-	-	-	-	-	-	-	
AppleQUIC		VHDCS	VHDCS		VHDCS 3	VHDCS	VHDCS	VHDCS 3	VHDCS	VHDCS 3		VHDCS		VHDCS	
quic-go															
Quinn		**VHDCRZS U**	**VHDCRZS BU**		**VHDCRZS BU**	**VHDCRZS**	VHDCS	**VHDCRZS 3**		VHS				VHDCRZS BUP	
Pandora		VHDC													
f5_test		VHDCS B	VHDCS B		VHDCS 3	V	VHDCS B	**VHDCS B3**	VHDCS	**VHDCS B3**				V	
Spindump															

Version Negotiation — Handshake — Stream Data — Connection Close — Resumption — 0-RTT — Stateless Retry

Table of contents

- QUIC: problem statement, concepts and process
- **Quinn: design, implementation status, future**

History

- Started Quinn last year in March
    - Subject of a RustFest May 2018 presentation
    - Found that Benjamin Saunders (@Ralith) had also started
        - Merged implementations in Q4, rapidly iterated
- By end of 2018, nicely conforming updated impl
- Released 0.2 release on Jan 21

Goals

- Implement the specifications faithfully
- Have clear abstractions & concepts
- Futures-based architecture — no polling
- Leverage the type system
- Good test coverage at different layers (currently ~77%)

quinn: futures-based application API

quinn-proto: deterministic protocol logic

Endpoint: dispatching datagrams to connections

Connection: core per-connection protocol logic

Crypto: TLS and encryption

Packet: packet encoding and decoding

Stream: streams state machine tracking API

quinn-h3

QPACK, HTTP 3

rustls and *ring*

- Highly secure, relatively easy to work with/hack on
- Added QUIC TLS needs to rustls:
  - Transport parameters extension
  - Handshake message-based interface
- *ring* algorithms are used directly after handshake
- Behind TLS interface traits to make it pluggable

CryptoSession trait

```rust
pub trait CryptoSession {
    fn alpn_protocol(&self) -> Option<&[u8]>;
    fn early_crypto(&self) -> Option<Crypto>;
    fn early_data_accepted(&self) -> Option<bool>;
    fn is_handshaking(&self) -> bool;
    fn read_handshake(&mut self, buf: &[u8]) -> Result<(), TransportError>;
    fn sni_hostname(&self) -> Option<&str>;
    fn transport_parameters(&self) -> Result<Option<TransportParameters>, TransportError>;
    fn write_handshake(&mut self, buf: &mut Vec<u8>) -> Option<Crypto>;
}
```

futures, tokio, h2

- Took API inspiration from h2 implementation
    - How to create a good futures-based API?
        - How does the API layer interact with library internals?
- Clean layering between API and protocol logic
- Possible future: tokio::net::quic

Ecosystem

- 11 others have contributed to Quinn so far
- In the process of improving Quinn, also contributed to:
  - libc (related to ECN support)
  - tokio
  - *ring*
  - rustls

Implementation status

- Very conformant draft 18 implementation
- Handles 70 MB/s with 128 kb streams (unoptimized)
- Complete QPACK implementation, not much H3 yet
- async/await implementation for testing on nightly branch
- Pace in transport/tls drafts seems to have slowed down

Open projects

- In progress:
  - H3
  - Task per connection
  - Finish pluggable TLS
- Open:
  - Stream prioritization
  - MTU discovery

quiche

- https://github.com/cloudflare/quiche
- Started late last year to replace their C++ version
- Uses BoringSSL bindings for TLS
- Low-level protocol API only

Request for contributions

- https://github.com/djc/quinn
- Your contribution is much appreciated
- There are a few issues in the repository
- Happy to mentor anyone to help get started