Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Advanced Computer Networking (ACN)

IN2097

Prof. Dr.-Ing. Georg Carle

Benedikt Jaeger, Johannes Zirngibl

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

General Remarks

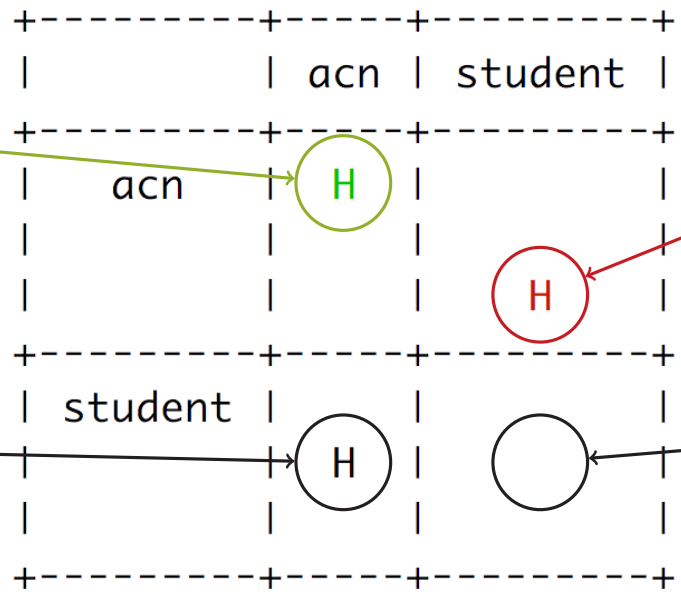
- Use the Moodle forum for discussion
- Compliance check of your implementations
  - If given a random test case, the client and server need to terminate with code 127
  - and print *"exited with code 127"*
- Binaries need to be executable outside the CI as well
  - We will test your implementations against each other and on real hardware
- The handshake for the first test case needs to be without a retry (see Moodle for more information)
- We use HTTP/3 and therefore h3 as ALPN
- For Problem2, all tests in the CI should be successful:

Success
Client and server application exited with error code 0 and the test results were verified.

Error
Client and server application exited with error code > 0 or the test was verified as false.

Unsupported
The client or server implementation did not support the given testcase meaning returning with exit code 127.

Not Compliant
The client or server implementation was not compliant. For example, they are missing or do not return with 127 on a random testcase.

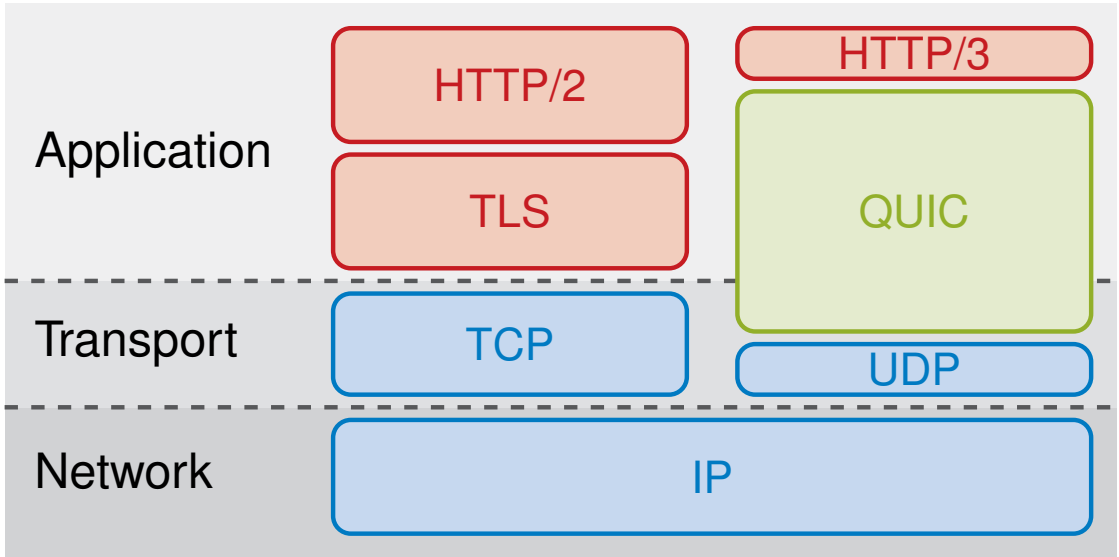


Project - Problem 1

1a)

- Introduced by Google and Jim Roskind
- First draft from 08.07.2016[1]
- While QUIC was an acronym initially, it is not used as such anymore

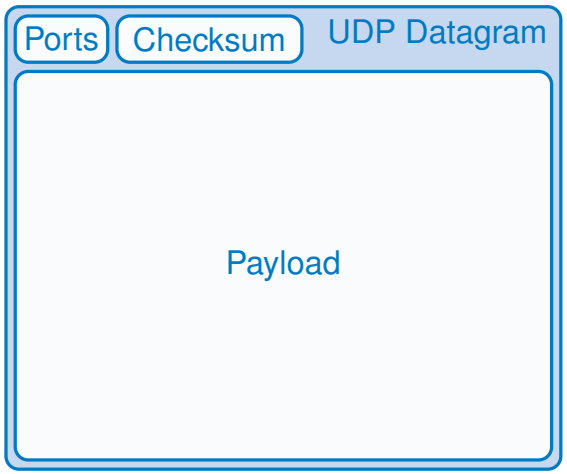
1b)

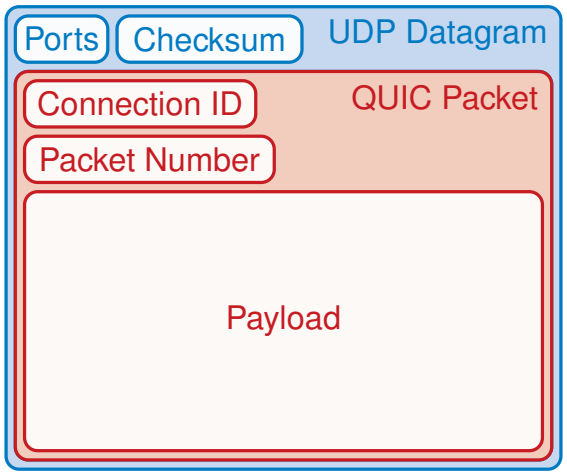


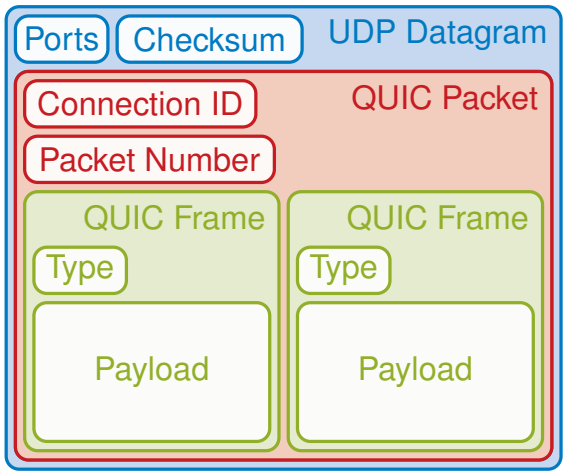
[1] https://datatracker.ietf.org/doc/html/draft-hamilton-quic-transport-protocol-00

Project - Problem 1

1c)







1d)

- Unilaterally declared parameters by each endpoint
- 17 different parameters exist in the specification
- Peers can e.g., set the initial size of the flow control window for the connection or streams, the maximum number of allowed streams, and options regarding connection migration
- Transport parameters **can** be send during the TLS handshake to authenticate values but can be **set** or **updated** later as well

1f)

Client:

- The client can send data via a QUIC stream (similar to TCP: in order, reliable, and stream-oriented)
- QUIC splits the stream into chunks which are encoded as frames
- The frames are combined into QUIC packets, multiple frames can be included in one packet
- The whole QUIC packet is encrypted and header protection is applied
- The encrypted QUIC PDU is passed to the UDP interface, e.g. using `send()`

Server:

- A UDP datagram arrives on the bound UDP socket
- The connection ID is parsed and the packet can be assigned to a connection
- The header protection is removed and the payload is decrypted using the connection's session keys
- The frames are parsed and the data stream is reconstructed from the data chunks using the offset field
- The server application can read from that stream interface, as with TCP, and, for example, respond to the received data in the same stream

Project - Problem 1

Selected Libraries

1e,g,h)

Library	Versions	CC Algorithms	Students
aioquic	v1, draft-[29-32]	NewReno	8
lsquic	v1, draft-[29,27]	Cubic, BBR, adaptive	4
quic-go	v1, draft-29	Cubic	3
quinn	v1, draft-[29-34]	Reno, Cubic, BBR	3
quiche	v1, draft-[27-29]	Reno, Cubic	1
haskell quic	v1, draft-29	RFC9002 pseudo code	1

Project - Problem 3

We will publish the description and additional information as Moodle post next Tuesday

- Start: December 21, 2021, 4:00 PM
- End: January 18, 2022, 4:00 PM
- Tasks: Implement additional test cases, e.g.:
  - Conduct multiple handshakes
  - Support retries
  - Configure transport parameters
- We will test implementations against each other and verify test cases