Optimizing the Host Protocol Stack

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Sandip Chakraborty

Mainack Mondal

Optimizations in Network Protocols

- **Offloading and Acceleration**
  - *TCP Offload Engine (TOE)*: Offloads TCP processing to the network interface card (NIC)
  - *Checksum and Segmentation Offload*: NICs handle checksum calculations and segmentation (breaking down large packets) rather than the CPU
  - *RDMA (Remote Direct Memory Access)*: Allows direct memory access between computers over a network, bypassing the CPU and TCP/IP stack for low-latency and high-throughput data transfer

Optimizations in Network Protocols

- **Offloading and Acceleration**
  - *TCP Offload Engine (TOE)*: Offloads TCP processing to the network interface card (NIC)
  - *Checksum and Segmentation Offload*: NICs handle checksum calculations and segmentation (breaking down large packets) rather than the CPU
  - *RDMA (Remote Direct Memory Access)*: Allows direct memory access between computers over a network, bypassing the CPU and TCP/IP stack for low-latency and high-throughput data transfer

- **Application Layer Optimizations**
  - *HTTP/2 and HTTP/3 (QUIC):* Flow multiplexing, use of UDP instead of TCP
  - *Content Caching and Delivery Network (CDN):* Content is cached closer to the user in geographically distributed locations, reducing load times and latency

Optimizations in Network Protocols

- **Packet Scheduling and Prioritization**
  - *Fair Queuing:* Ensures each connection receives an equitable share of the bandwidth, preventing any single application from monopolizing the network
  - *Active Queue Management (AQM):* Techniques like **RED (Random Early Detection)** and **CoDel (Controlled Delay)** reduce latency by managing queues at network routers and switches, preventing congestion before it builds up.

Optimizations in Network Protocols

- **Packet Scheduling and Prioritization**
  - *Fair Queuing:* Ensures each connection receives an equitable share of the bandwidth, preventing any single application from monopolizing the network
  - *Active Queue Management (AQM):* Techniques like **RED (Random Early Detection)** and **CoDel (Controlled Delay)** reduce latency by managing queues at network routers and switches, preventing congestion before it builds up.

- **Latency Reduction Techniques**
  - *Edge Computing:* Processing data closer to the data source reduces latency and offloads the core network.
  - *DNS Caching and Prefetching:* Reduces DNS resolution time by caching responses and prefetching likely-needed addresses.
  - *TCP BBR (Bottleneck Bandwidth and RTT):* Adapts the TCP congestion window based on real-time measurements of available bandwidth and round-trip time (RTT)

TCP Offload Engine (TOE)

Improving TCP performance by offloading their functionalities to the NIC

Offloading TCP Functionalities

- TCP processing tasks
  - connection establishment
  - Acknowledgment
  - packet ordering
  - Retransmission
  - congestion control
  - teardown
- CPU handles TCP processing in the traditional TCP/IP stack
  - The tasks are heavy CPU intensive, particularly at high data rate

CPU Usage by TCP

Connection setup/ teardown	TCP processing and state update	24.0%	60.5%
	TCP connection state init/destroy	17.2%	
	Packet I/O (control packet)	10.2%	
	L2-L3 processing/forward	9.1%	
Message delivery	TCP processing and state update	11.0%	29.0%
	Message copy via socket buffer	8.4%	
	Packet I/O (data packet)	5.1%	
	L2-L3 processing/forward	4.5%	
Socket/epoll API calls			5.6%
Timer handling and context switching			3.5%
Application logic			1.4%

CPU usage breakdown of a user-level TCP echo server (a single 64B packet exchange per connection)

Source: https://www.usenix.org/system/files/nsdi20-paper-moon.pdf

Offloading TCP Functionalities

- TCP processing tasks
  - connection establishment
  - Acknowledgment
  - packet ordering
  - Retransmission
  - congestion control
  - teardown
- CPU handles TCP processing in the traditional TCP/IP stack
  - The tasks are heavy CPU intensive, particularly at high data rate
- With a TOE-enabled NIC, these TCP tasks are performed directly on the NIC, allowing the CPU to focus on application processing

Works on TCP Offload

AccelTCP: Accelerating Network Applications with Stateful TCP Offloading

YoungGyoun Moon and SeungEon Lee, *KAIST;* Muhammad Asim Jamshed, *Intel Labs;* KyoungSoo Park, *KAIST*

https://www.usenix.org/conference/nsdi20/presentation/moon

FlexTOE: Flexible TCP Offload with Fine-Grained Parallelism

Rajath Shashidhara, *University of Washington;* Tim Stamler, *UT Austin;* Antoine Kaufmann, *MPI-SWS;* Simon Peter, *University of Washington*

https://www.usenix.org/conference/nsdi22/presentation/shashidhara

Rearchitecting the TCP Stack for I/O-Offloaded Content Delivery

Taehyun Kim and Deondre Martin Ng, *KAIST;* Junzhi Gong, *Harvard University;* Youngjin Kwon, *KAIST;* Minlan Yu, *Harvard University;* KyoungSoo Park, *KAIST*

https://www.usenix.org/conference/nsdi23/presentation/kim-taehyun

TCP Offload through Connection Handoff

Hyong-youb Kim and Scott Rixner
Rice University
Houston, TX 77006
{hykim, rixner}@rice.edu

Autonomous NIC Offloads

Boris Pismenny
Technion and NVIDIA
Haifa, Israel

Haggai Eran
Technion and NVIDIA
Haifa, Israel

Aviad Yehezkel
NVIDIA
Yokne'am Illit, Israel

Liran Liss
NVIDIA
Yokne'am Illit, Israel

Adam Morrison
Tel Aviv University
Tel Aviv, Israel

Dan Tsafrir
Technion and VMware Research
Haifa, Israel

The Idea of TCP Offload is Not New !!

- NIC developer have thought about TCP offload long back
  - TCP has been one of the heaviest protocols in the TCP/IP architecture

- However, it was not very successful !!
  - Has been "**a solution in search of a problem**" for several decades

TCP offload is a dumb idea whose time has come

Jeffrey C. Mogul
Hewlett-Packard Laboratories
Palo Alto, CA, 94304
JeffMogul@acm.org

HotOS 2023

Why TCP Offload Was Unsuccessful in the Past?

- Processing TCP headers should not take many cycles
  - Efficient methods are available, like "header prediction" by Jacobson (https://www.freesoft.org/CIE/RFC/1323/17.htm)

- Adding a transport protocol implementation to a NIC requires considerably more hardware complexity than a simple MAC-layer-only NIC
  - TOE can become the performance bottleneck (Moore's law says CPU will continue be faster and thus, NIC might fall behind)

- TOE interfaces needed additional bus header (to inform the NIC about the incoming TCP packets), and thus makes the interface complex
  - May introduce more overhead

Why TCP Offload Was Unsuccessful in the Past?

- The TOE must maintain connection state for each TCP connection, and must coordinate this state with the host operating system.
  - Especially for short-lived connections, any savings gained from less host involvement in processing data packet is wasted by this extra connection management overhead.

- If the transport protocol resides in the NIC, the NIC and the host OS must coordinate responsibility for resources such as data buffers, TCP port numbers, etc.
  - The ownership problem for TCP buffers is more complex than the seemingly analogous problem for packet buffers, because outgoing TCP buffers must be held until acknowledged, and received buffers sometimes must be held pending reassembly.

Why TCP Offload Was Unsuccessful in the Past?

- Several deployment-related issues
  - Some servers must maintain huge number of connections
  - Patching TOE at the NIC is much more difficult than patching the host OS kernel
  - Use of TOE increases testing complexity for the hardware vendors
  - Detecting the source of a bug (OS or NIC) becomes much harder (debugging hardware is complex)
  - Designing management interface is difficult – network administrators need to trace the functionalities and often configure parameters

Ethernet as the Storage Interconnector

- Apart from networks, computers generate high data rate on three kinds of channels
  - Graphics Systems
  - Storage Systems
  - Interprocessor interconnects
- Historically, these rates have been supported by special purpose interface hardware (SCSI, Fiber Channel, etc.)
  - Induces cost and reduces flexibility (becomes vendor-specific)
- However, with cheaper Gbps Ethernet the cost can be reduced, and more flexibility can be provided in the interconnection fabrics

Challenges with using Ethernet as the Interconnector

- Tradition network stack uses a large amount of data copy
  - read() and write() -- allow applications to decide when and how data buffer appears in their address space
- Copy overhead affects the storage performance significantly
  - There are software-based approaches came up for copy-avoidance
- However, software-based copy avoidance has several issues
  - If MSS < VM Page Size (which is often the case), page remapping is costly
  - Needs new APIs, compatability issues with legacy OSes
- A prominent solution widely deployed: **Remote Direct Memory Access (RDMA)**

Remote Direct Memory Access (RDMA)

- Allows data to be transferred directly between the memory of two computers over a network
  - Does not involve the CPU, OS, or other components on either system

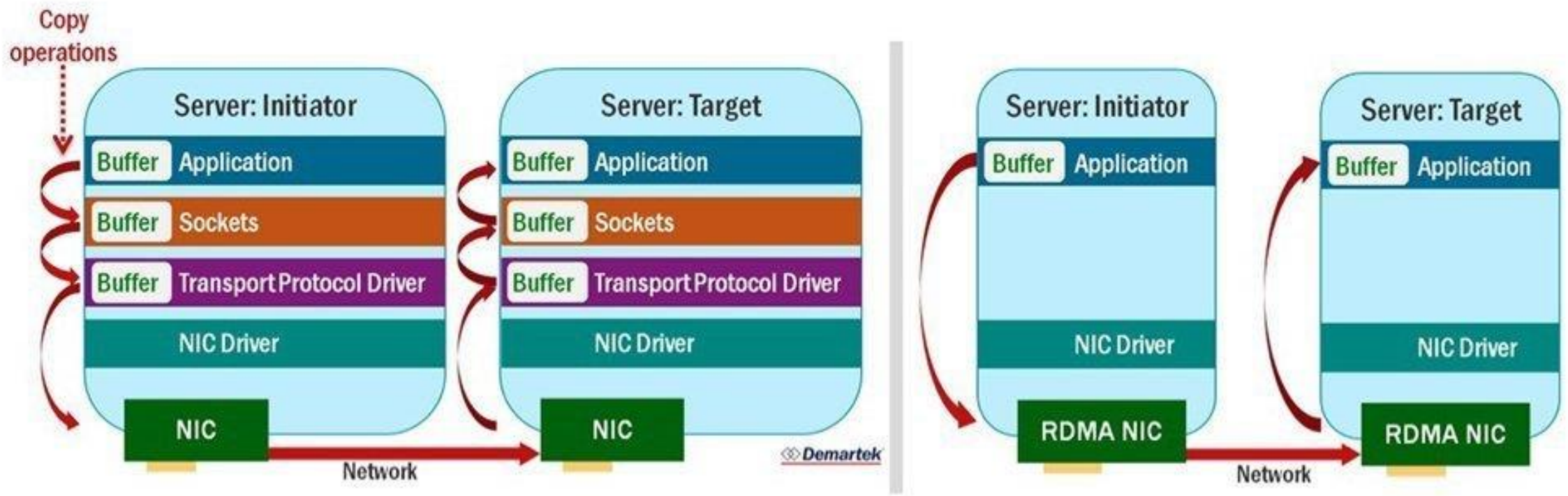


Image Source: https://www.techtarget.com/searchstorage/definition/Remote-Direct-Memory-Access

RDMA-Supported Protocols

- **InfiniBand**: A high-speed network
  - Used in supercomputing and HPC
  - Has built-in support for RDMA, providing low-latency and high-throughput communication.

- **RDMA over Converged Ethernet (RoCE)**: Allows RDMA to work over standard Ethernet networks
  - **RoCEv1**: Requires a lossless Ethernet network.
  - **RoCEv2**: Works over regular IP networks with additional features for improved scalability.

- **iWARP**: A protocol that allows RDMA to work over standard TCP/IP networks

NIC with RDMA

- RDMA sidesteps the problems with software-based copy-avoidance schemes
  - NIC Hardware implements the RDMA protocol
  - Kernel or application software registers buffer regions via the NIC driver, obtains protected buffer reference tokens (called *region IDs*)
  - Software exchanges these region IDs with its connection peer, via RDMA messages sent over the transport connection
  - Special RDMA message directives ("verbs") enable a remote system to read or write memory regions named by the region IDs
  - The receiving NIC recognizes and interprets these directives, validates the region IDs, and performs protected data transfers to or from the named regions

NIC with RDMA

- An RDMA-enabled NIC (RNIC) needs its own implementation of all lower-level protocols
- RDMA aims to substitute for hardware storage interfaces
  - Must provide highly reliable data transfer
  - Must be layered over a reliable transport such as TCP or SCTP

NIC with RDMA

- An RDMA-enabled NIC (RNIC) needs its own implementation of all lower-level protocols
- RDMA aims to substitute for hardware storage interfaces
  - Must provide highly reliable data transfer
  - Must be layered over a reliable transport such as TCP or SCTP

Offloading the transport layer becomes valuable not for its own sake, but rather because that allows offloading of the RDMA layer

NIC with RDMA

- An RDMA-enabled NIC (RNIC) needs its own implementation of all lower-level protocols
- RDMA aims to substitute for hardware storage interfaces
  - Must provide highly reliable data transfer
  - Must be layered over a reliable transport such as TCP or SCTP

Offloading the transport layer becomes valuable not for its own sake, but rather because that allows offloading of the RDMA layer

RDMA applications are likely to use a relatively small number of low-latency, high-bandwidth transport connections, precisely the environment where TCP offloading might be beneficial

TOE Over RNIC Has Challenges!

- **Getting Semantics Right:** RDMA introduces many issues related to buffer ownership, operation completion, and errors
  - **OS-to-RDMA Interfaces:** These interfaces include, for example, buffer allocation; mapping and protection of buffers; and handling exceptions beyond what the RNIC can deal with (such as routing and ARP information for a new peer address).
  - **Applications to RDMA Interfaces:** These interfaces include, for example, buffer ownership; notification of RDMA completion events; and bidirectional interfaces to RDMA verbs.

- **Network Configuration and Management:** RNICs will require IP addresses, subnet masks, etc., and will have to report statistics for use by network management tools.
  - OS should provide a “single system image” for network management functions

TOE Over RNIC Has Challenges!

- **Defense Against Attacks:** introduces several tricky problems, especially in the area of security
  - Offloading the transport protocol exacerbates the security problem by adding more opportunities for bugs (IPSec may not be able to secure IP anymore)
  - RDMA bug could be much more severe than traditional protocol-stack bugs, because it might allow unbounded and unchecked access to host memory

AccelTCP: Accelerating Network Applications with Stateful TCP Offloading

YoungGyoun Moon and SeungEon Lee, *KAIST;* Muhammad Asim Jamshed, *Intel Labs;* KyoungSoo Park, *KAIST*

https://www.usenix.org/conference/nsdi20/presentation/moon

NSDI 2020

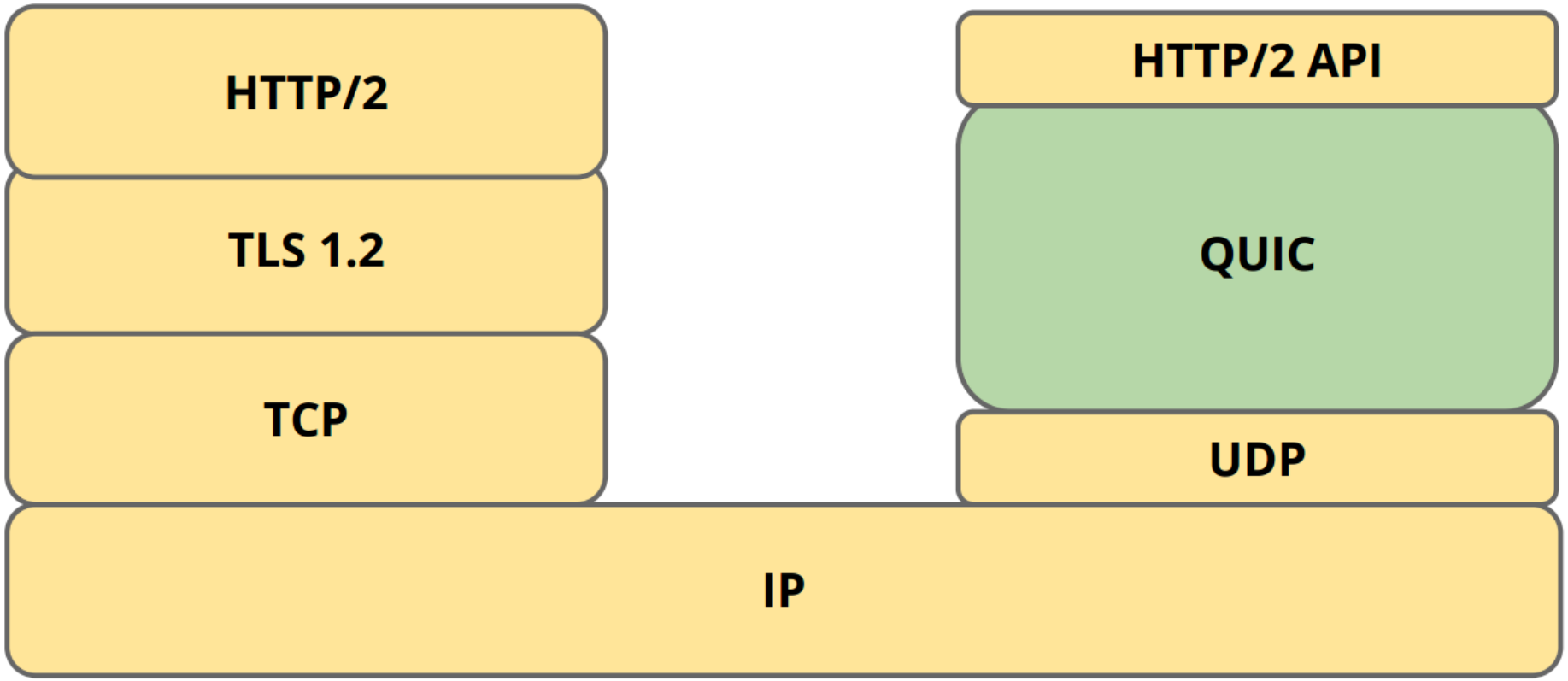
https://www.usenix.org/sites/default/files/conference/protected-files/nsdi20_slides_moon.pdf

The QUIC Transport Protocol: Design and Internet-Scale Deployment

Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, Zhongyi Shi *
Google
quic-sigcomm@google.com

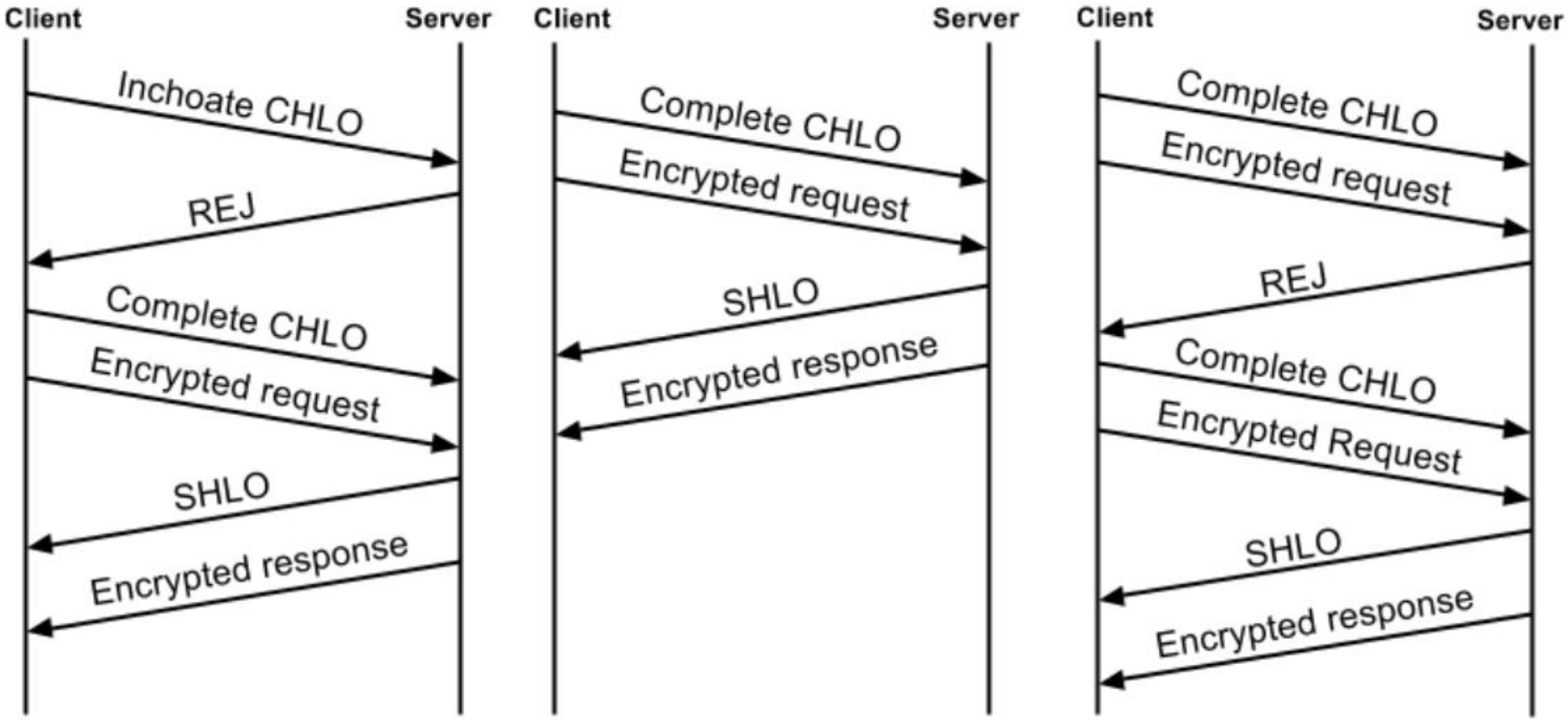
SIGCOMM 2017

The QUIC Stack



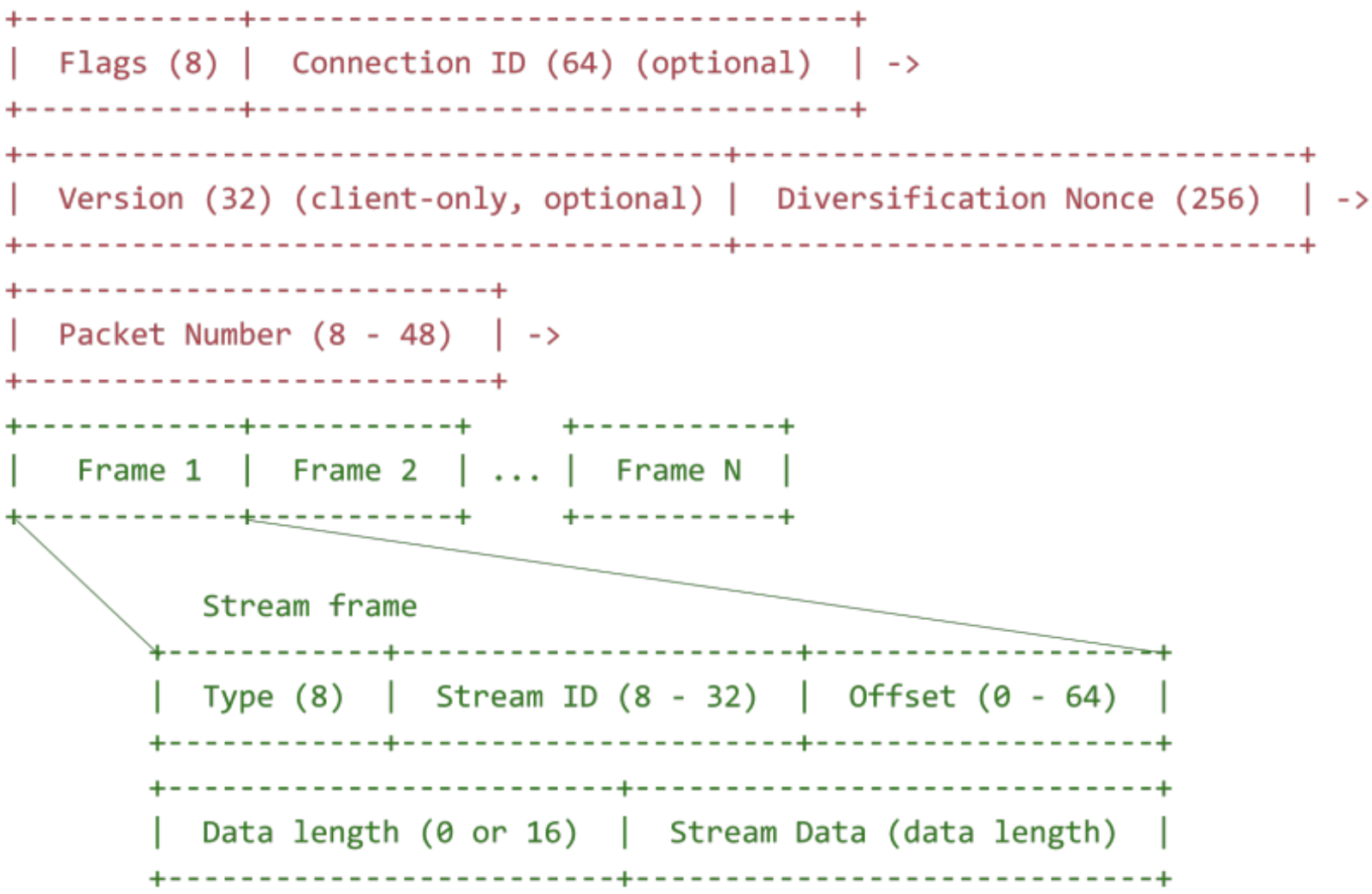
https://datatracker.ietf.org/doc/html/rfc9000

Connection Establishment in QUIC



Initial 1-RTT Handshake **Successful 0-RTT Handshake** **Rejected 0-RTT Handshake**

Structure of a QUIC Packet



Some resources related to this topic