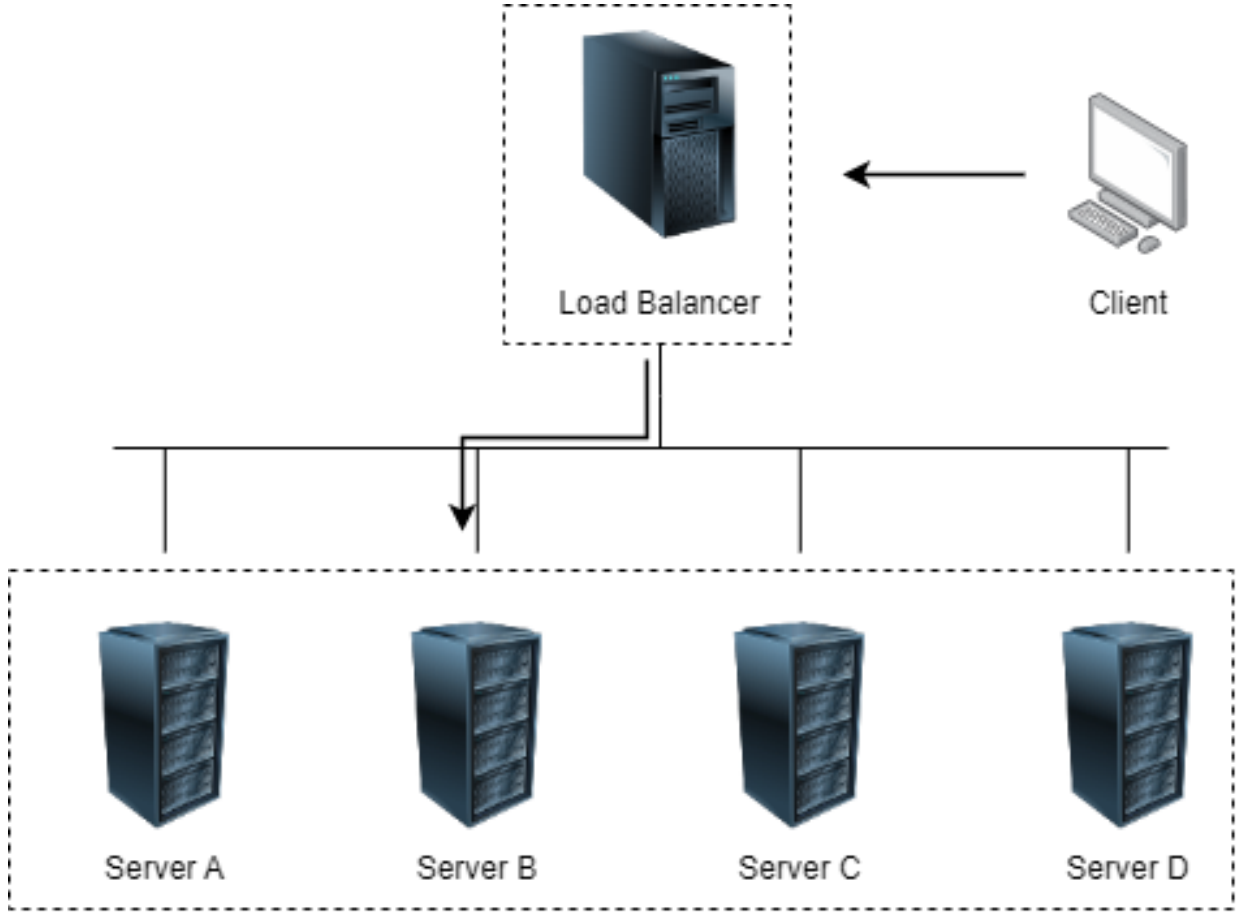
QDSR: Accelerating Layer-7 Load Balancing by Direct Server Return with QUIC

Ziqi Wei*, Zhiqiang Wang*, Qing Li✉, Yuan Yang, Cheng Luo, Fuyu Wang, Yong Jiang, Sijie Yang, Zhenhui Yuan

(*: co-first authors, ✉: corresponding author)

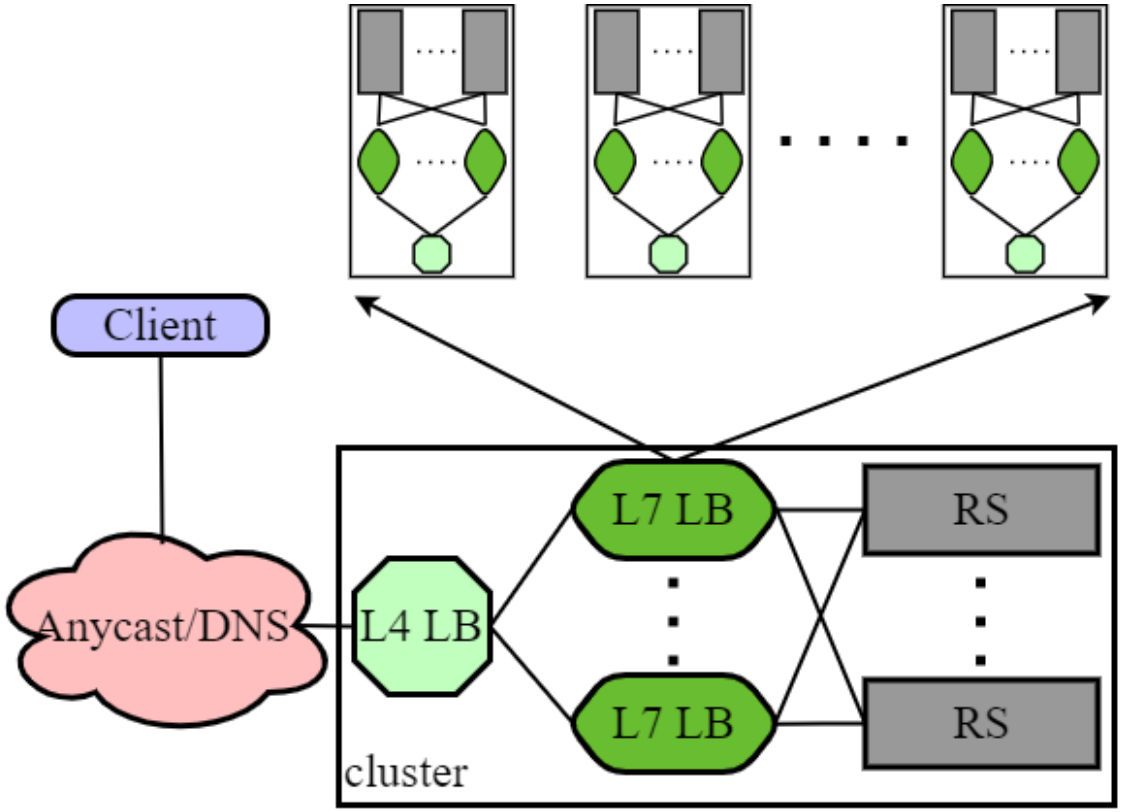
Load Balancing

- A service that distributes traffic to multiple backend real servers.
  - High availability
  - Elastic Scalability
  - Security and stability
  - Multi-protocol Forwarding
    - Layer-4 Load Balancing
    - Layer-7 Load Balancing



Layer-7 Load Balancing

- Usually used with L4 load balancing
- Support more flexible content-based scheduling (easily achieve sticky redirect)
- Multiple clusters are connected together through Layer-7 scheduling

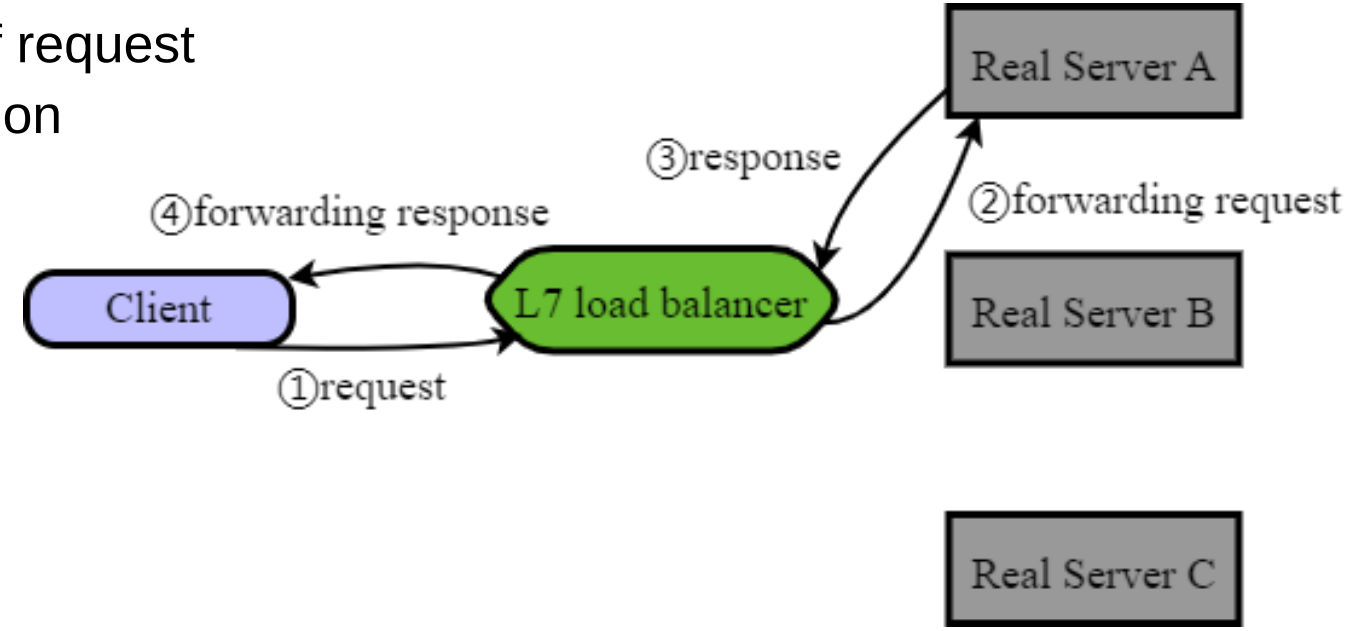


Layer-7 Load Balancing

- More diverse usage scenarios, blurring the boundaries between load balancers and real servers
- The servers in the cluster are both L7 load balancers and real servers

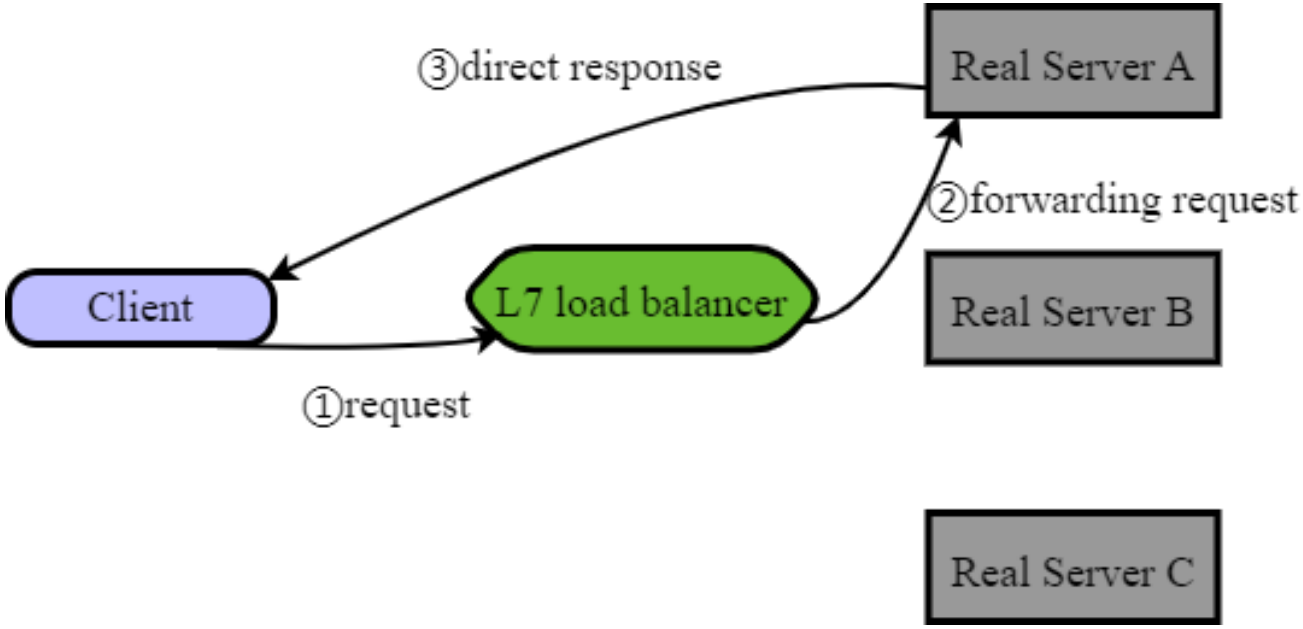
Layer-7 Load Balancing

- Advantages of layer-7 load balancing
  - Content-based scheduling
  - Fault tolerance (proactive health check)
  - Web application firewall (WAF)
- Disadvantages of traditional load balancing: redundant downlink forwarding
  - Additional performance overhead of the load balancer
  - Additional transmission delay of request
  - Additional bandwidth consumption
    - WAN: bandwidth fee
    - LAN: congestion possibility



Direct Server Return (DSR)

- Layer-4 direct server return: Linux Virtual Server (LVS) direct routing mode
  - Not support content-based scheduling
- Layer-7 direct server return: [USENIX NSDI ‘21] Prism
  - Achieve DSR by serial TCP connection and TLS state hand-off between L7 load balancer and real servers
  - Not support parallel scheduling, therefore, not applicable to HTTP/2 and QUIC



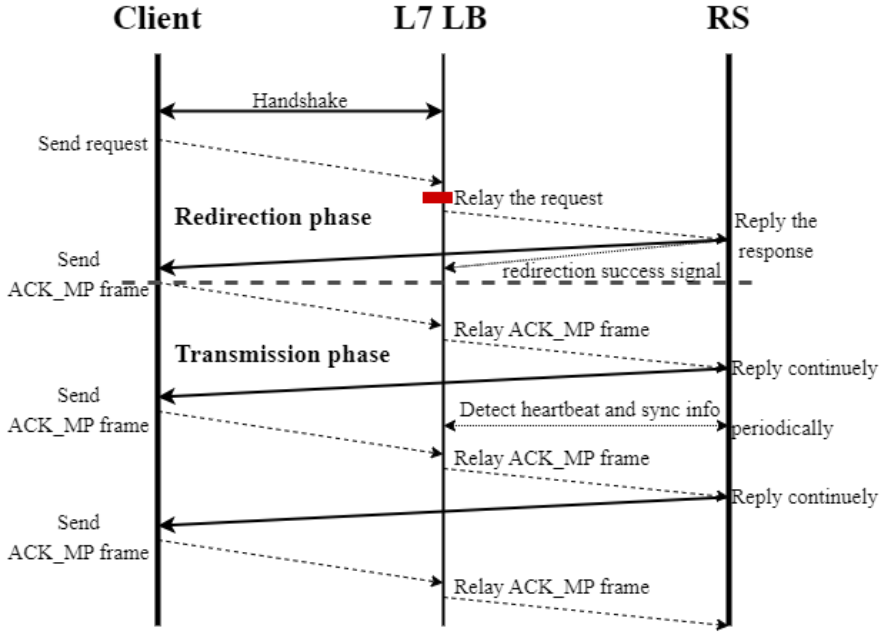
How to Support Direct Server Return in HTTP/3?

- QUIC is the transport layer protocol of HTTP/3
  - One HTTP request corresponds to one QUIC stream
  - Different QUIC streams are parallel and there is no head-of-line blocking

- Challenge
  - Migrate different QUIC stream to different real servers at the same time
  - Multiple real servers send HTTP responses without affecting each other

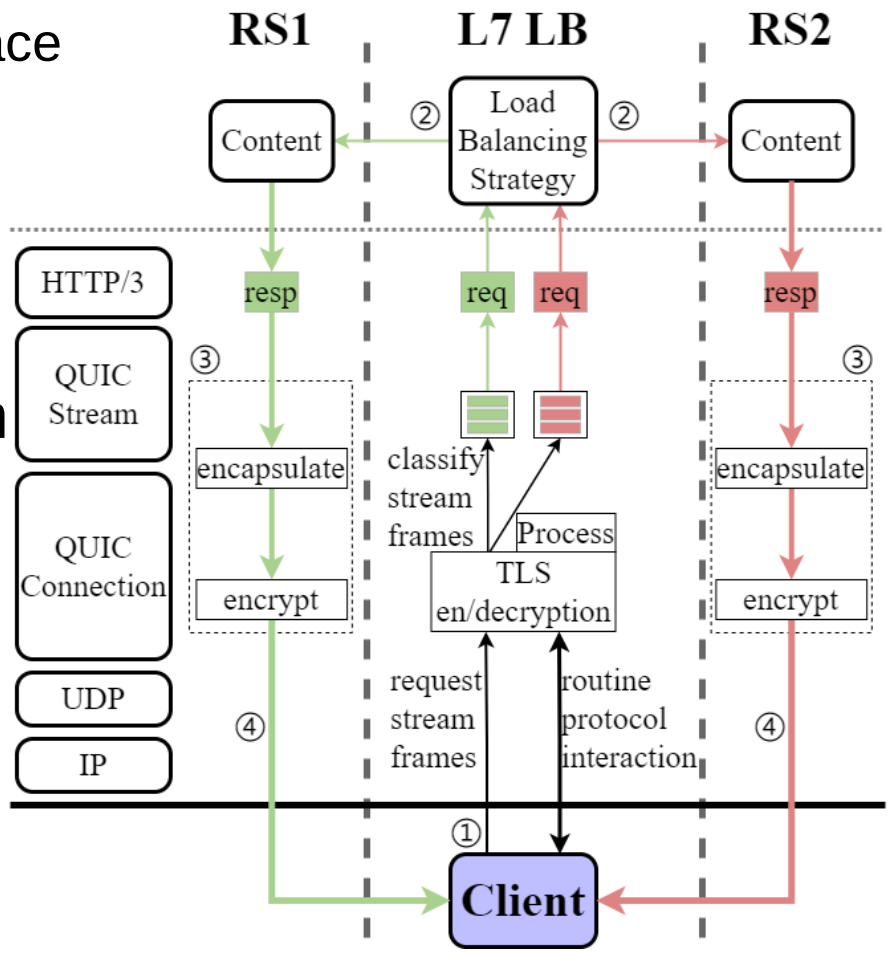
Two Phase Stream Hand-Off to Achieve Parallel DSR

- Redirection phase
  - Schedule: select one real server according to some strategies
  - Hand off QUIC stream: transfer QUIC stream state to the real server and use this stream direct response client
- Transmission phase
  - The real server sends a downstream response directly to the client
  - The load balancer forwards control messages from the client to different servers



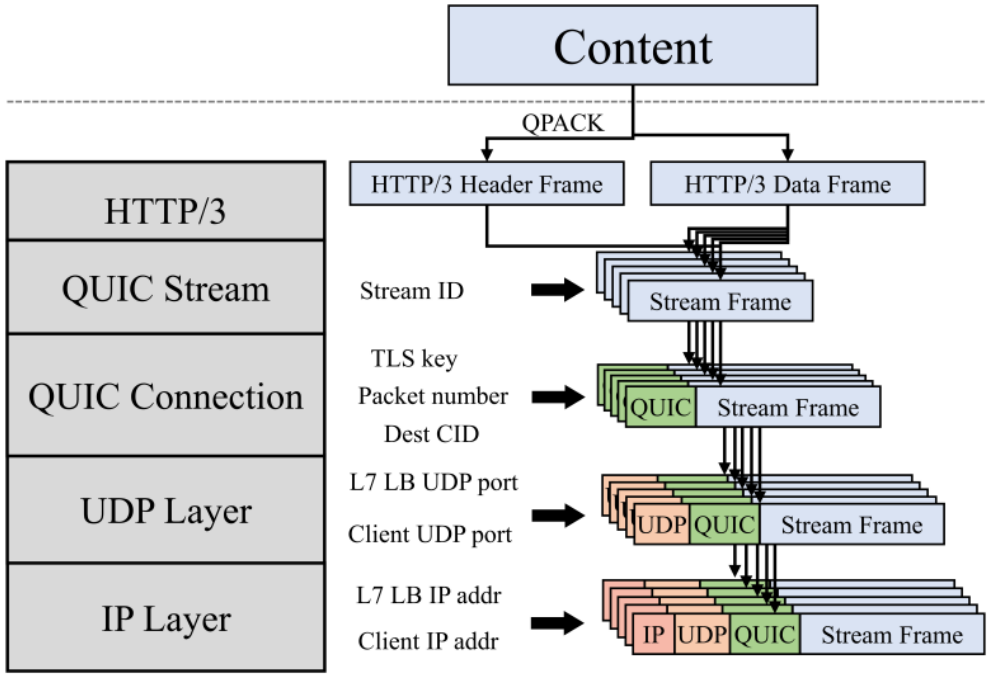
Redirection Phase

- Necessary state for QUIC stream hand-off to be synchronized to the real server
  - QUIC connection ID and it associated packet number space
  - QUIC stream ID and its flow control limitation
  - Negotiated TLS key
  - Client IP, Client Udp port
  - L7 LB IP (usually is virtual IP), L7 LB Udp port
- Real server recovers forged connection and stream
- Real server sends response directly to the client



Forged Packet by Real Servers

- Response header and content are encapsulated as HTTP/3 frames
- HTTP/3 frames will be split into QUIC stream frames
- QUIC stream frames should be encrypted and encapsulated as QUIC packets
- QUIC packets will be added with the forged IP address and UDP port



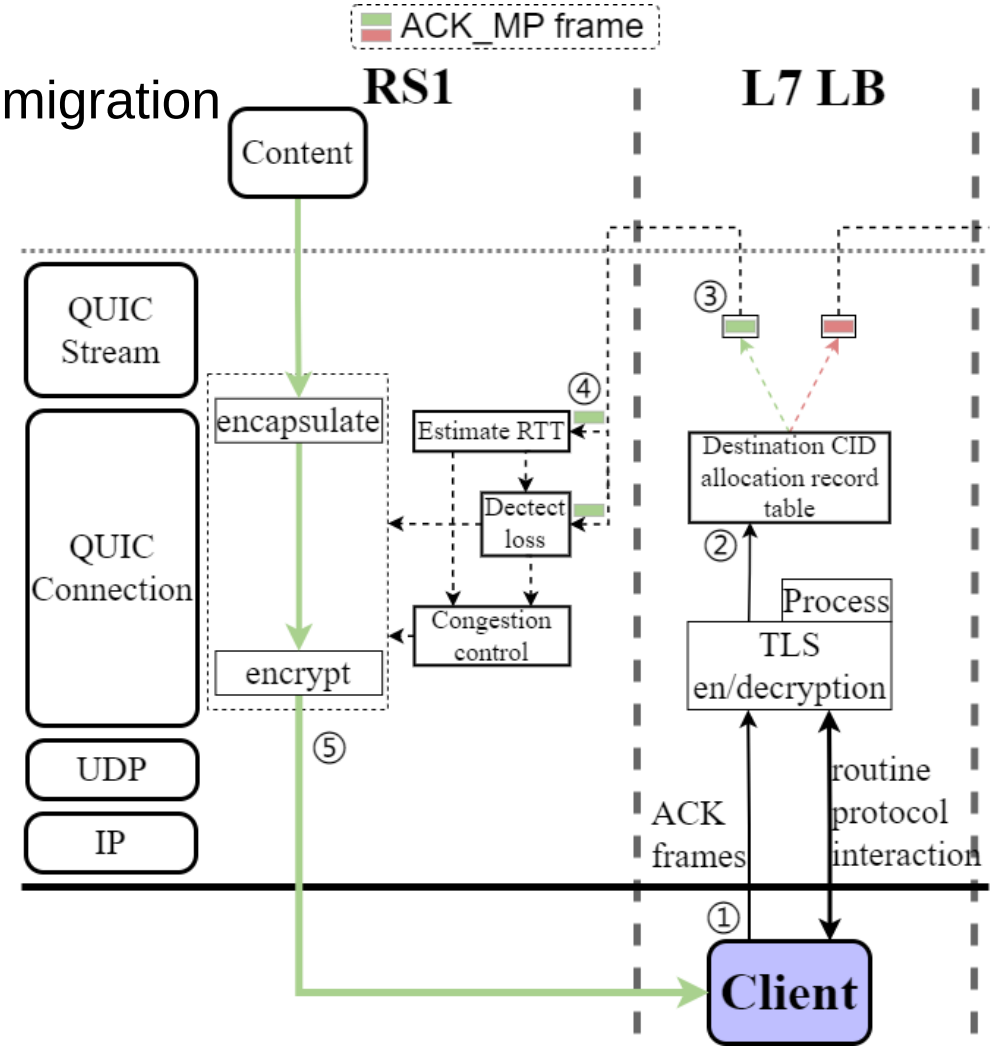
Packet Number Space Isolation

- IETF QUIC only supports one application packet number space
- One packet number space can’t support parallel multiplex sender
  - Clients receive QUIC packet not in the order of packet number
    - The client's fast packet loss detection mechanism fails
    - Replay packet attacks are difficult to recognized

- Introduce individual packet number space for every sender like multipath
  - Isolate transmission states of different senders
    - Congestion controll
    - Lost detect
    - TLS state
  - Compatible with multipath transmission

Transmission phase

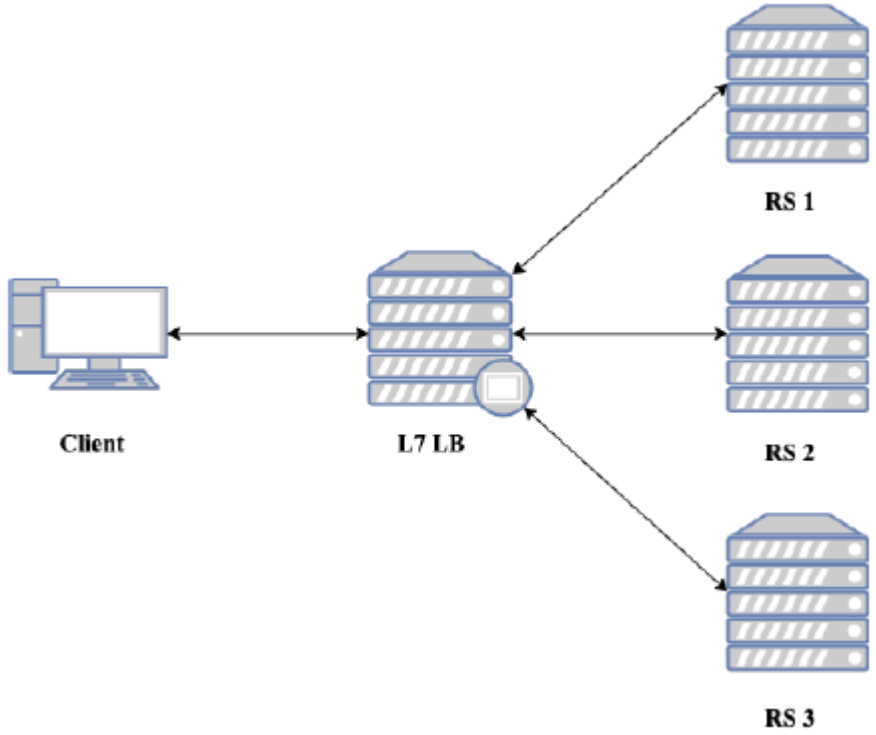
- Layer-7 load balancer: a control center to coordinate multiplex real servers
  - Forward ACK frames to the real server
  - Update state to real servers when QUIC connection migration
  - Update key to real server when TLS key updated
  - Other control information
- Real server as a transparent real sender
  - Lost detection and retransmission
  - Congestion control



Implement and Evaluation

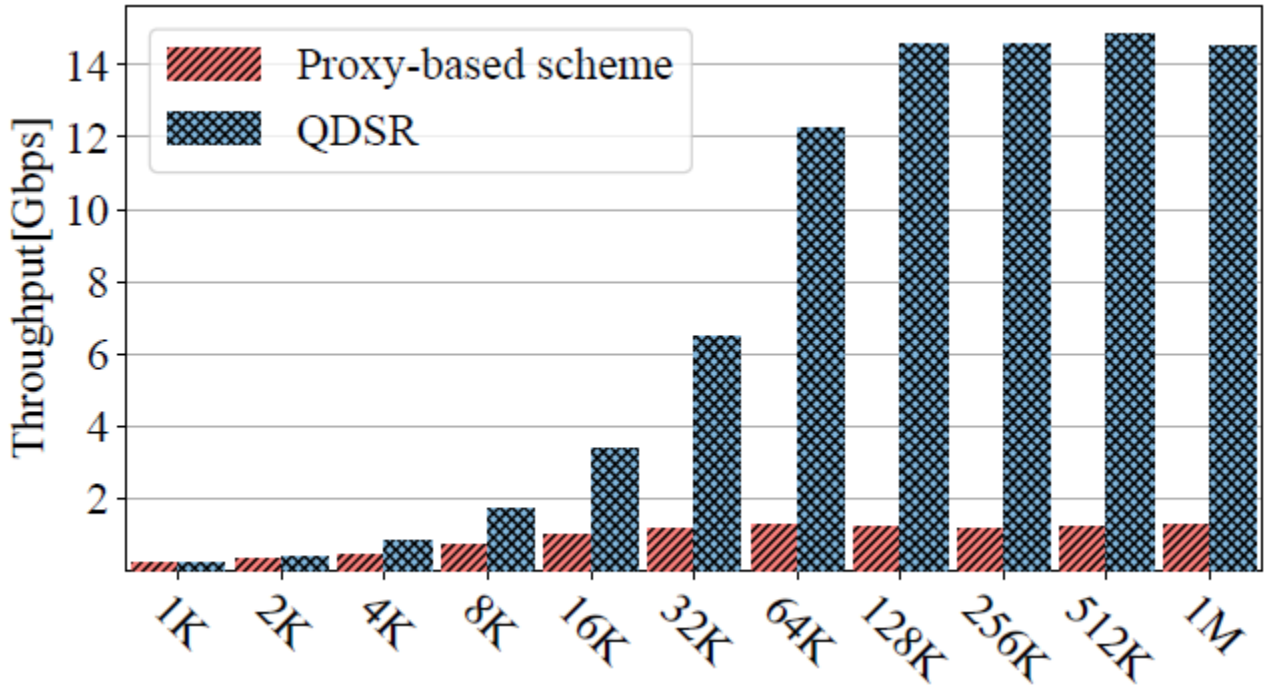
- Implement
  - Layer-7 load balancer: nginx
  - Real server: apache traffic server
  - Client: lsquic
- Baseline: Proxy-based scheme
- LAN testbed

Type	CPU	Memory	Link
A	Intel Xeon Silver 4114 2.2GHz, 20 cores	16 GB	40 Gbps
B	Intel Xeon Silver 4216 2.1GHz, 64 cores	256 GB	100 Gbps
C	Intel Xeon Silver 4208 2.1GHz, 32 cores	128 GB	100 Gbps



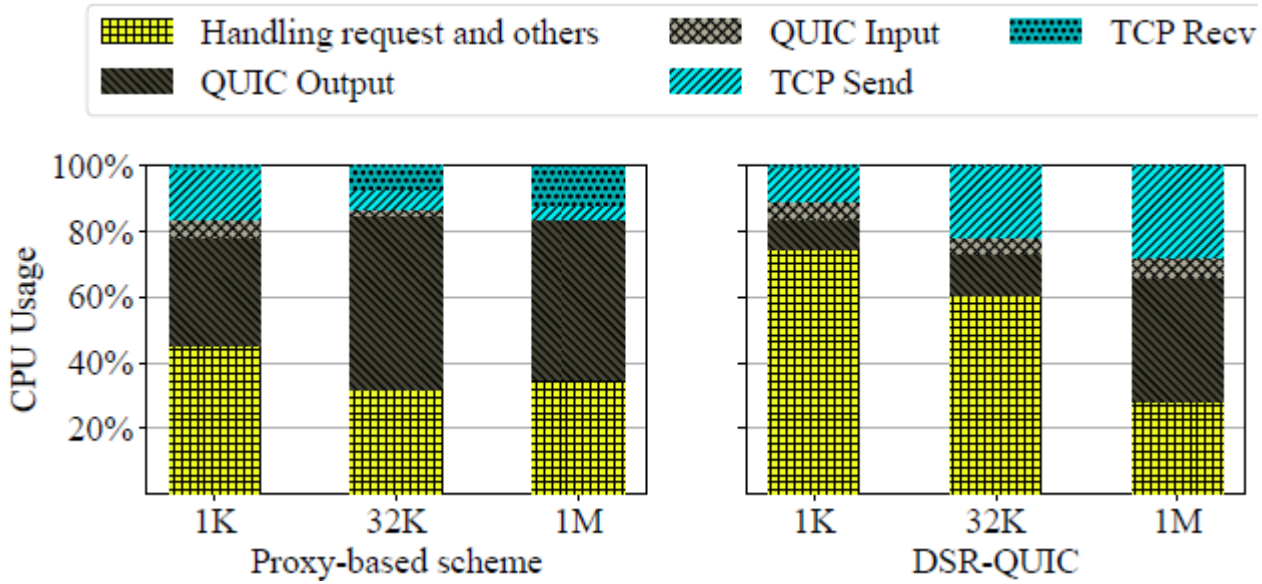
Single Core Throughput of Layer-7 Load Balancer

- Layer-7 Load Balancer: Single CPU core
- Enough real servers
- protocol between load balancer and real servers: TCP
- Concurrency: 8 QUIC connections and 8 streams multiplexing pre-connection



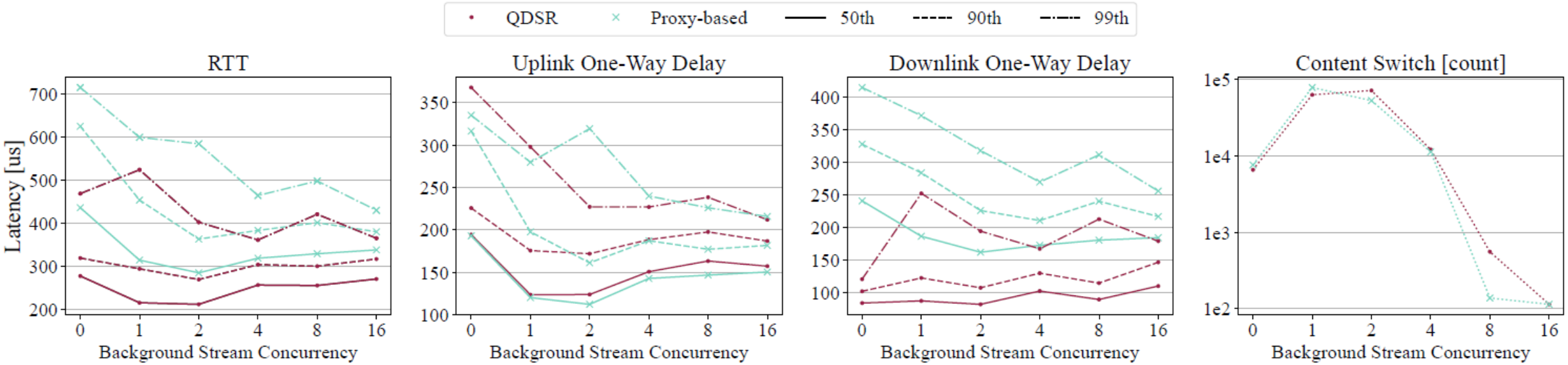
CPU Bottleneck

- Small requested objects: the bottleneck was processing requests
- Requested objects exceeds 16K: QPS decrease and relay consumption increase



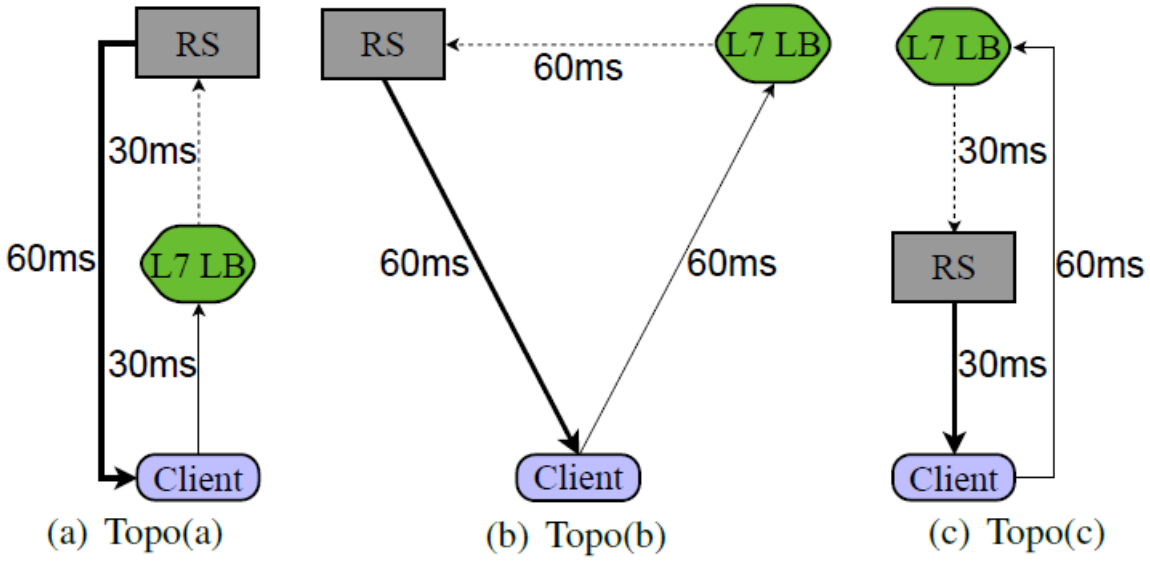
Performance in Lan

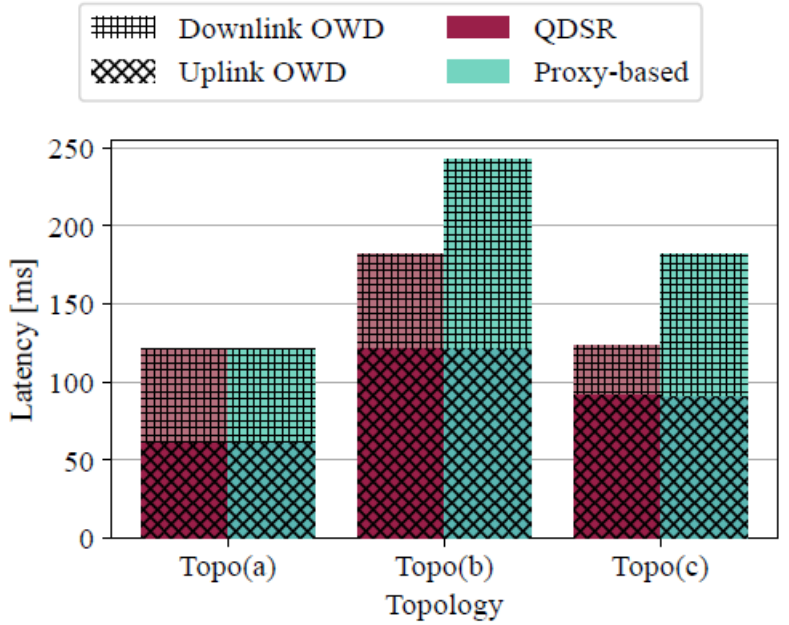
- Measured after clock synchronization
- Downlink one-way delay is significantly lower than proxy-based scheme
  - Reduce intranet traffic



End-to-End Delay over WAN

- Simulation experiments using mahimahi
- The transaction benefits from topology of direct server return





Conclusion

- Illustrate the significance of parallel direct server return for layer-7 load balancing
- Design QDSR to achieve parallel direct server return without damaging flexibility and scalability of layer-7 load balancing
- Implement QDSR and evaluate its performance compared with traditional proxy-based layer-7 load balancing. The results show that the performance is greatly improved

- Attention to our future work
  - TQUIC https://github.com/tencent/tquic
  - Tencent EdgeOne https://edgeone.ai
  - Other detail discussion: zhqiangwang@tencent.com

Thank you