Scheduling Diverse QUIC Streams

Yufeng Chen, Akhil Prasad, Sonia Fahmy, Voicu Popescu
Department of Computer Science, Purdue University
E-mail: {chen4044, prasad67, fahmy, popescu}@purdue.edu

Background

- Applications like VR have delay and loss-(in)sensitive data.
- QUIC offers multiple streams and easy application integration.
- Applications can use multiple reliable streams and unreliable datagrams in QUIC.
- Scheduling among them becomes important.

Challenge

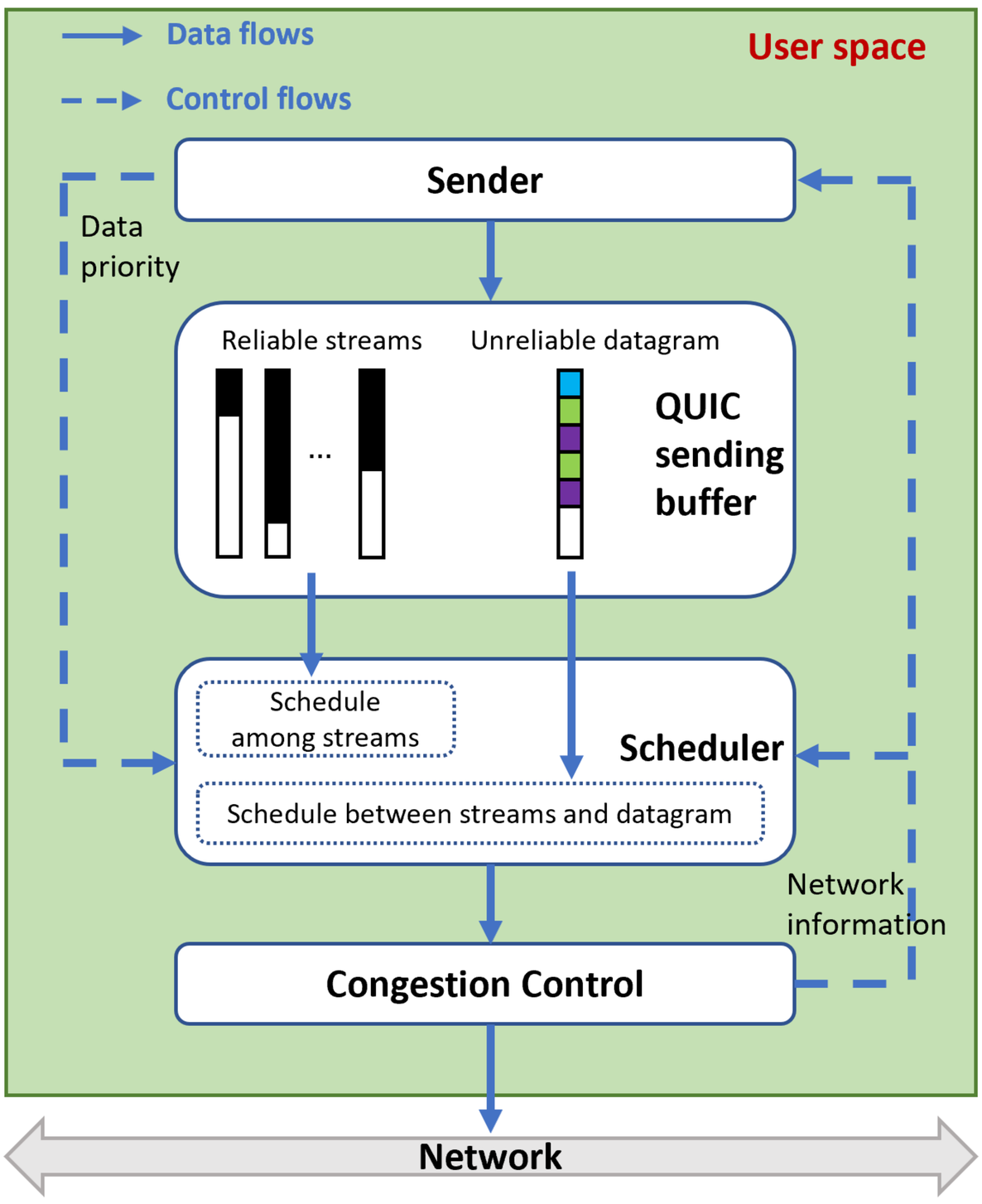
- Balance overhead and application Quality of Experience (QoE) as the number of concurrent streams, and the scheduling sophistication, increase.

	Static scheduler	Dynamic scheduler
Data structure	Priority Queue (priority levels) + Queue for each priority level	Queue
Implementation (Cloudflare quiche)	B-Tree + Double-ended vector	Double-ended vector
Enqueue operation	$O(\log m)$	$O(1)$
Dequeue operation	$O(1)$ If the priority level becomes empty, needs $O(\log m)$ to remove	$O(n)$

$\boldsymbol{m}$: current number of priority levels $\boldsymbol{n}$: current total number of streams

Design

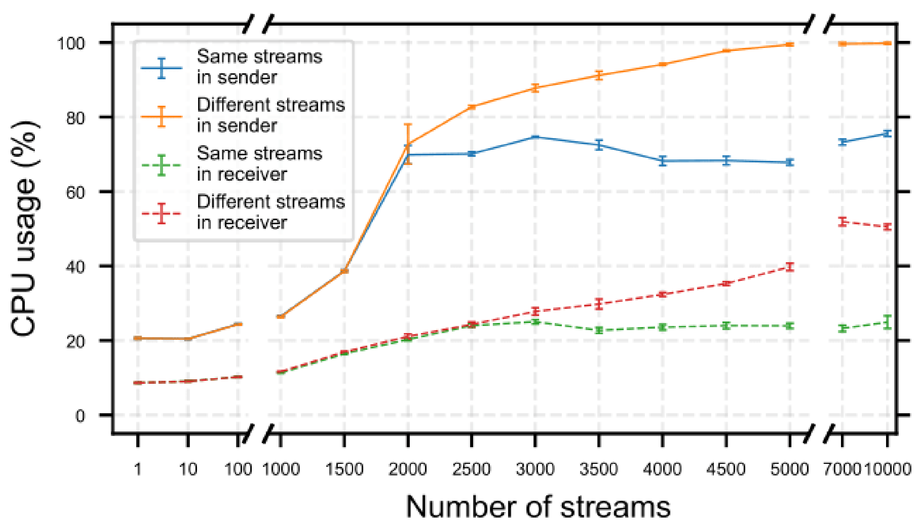
- Data properties and real-time network information can be used for scheduling at the sender.
- Schedulers can be static or dynamic, depending on the operations executed when dequeueing.



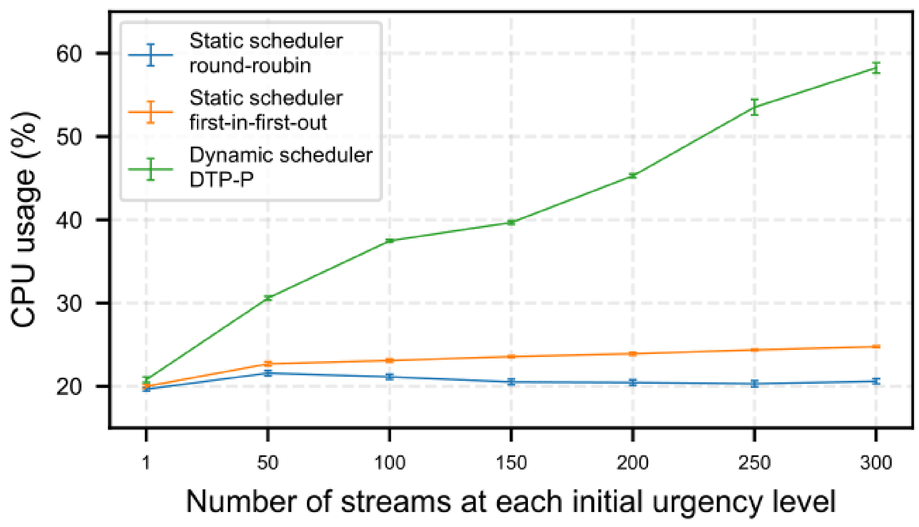
Methodology

- Measure resource usage, sender queueing delay (from data generation to sending onto network), and application QoE.

CPU Usage with concurrent streams



CPU usage with static/dynamic schedulers



Future Work

- Map diverse VR application data types to reliable streams and unreliable datagrams.
- Experiment with scheduling approaches.
- Test under different network conditions and QUIC implementations and parameters.

This work has been supported in part by NSF grant 2212200.