

No Silver Bullet: Extending SDN to the Data Plane

Anirudh Sivaraman, Keith Winstein, Suvinay Subramanian, and Hari Balakrishnan  
MIT Computer Science and Artificial Intelligence Laboratory, Cambridge, Mass.  
{anirudh, keithw, suvinay, hari}@mit.edu



The Data Plane is continuously evolving.

- New scheduling and queue management algorithms are proposed every year.
- Each wins in its own evaluation under its own workloads.
- There is a tacit belief in a final answer to these questions.

Yet, there is no silver bullet.

Different applications care about different things.

- Interactive video conferencing apps need both high throughput & low delay.
- Bulk transfers care only about high throughput.
- Web browsers care about minimizing flow completion time (FCT).

Applications run different transport protocols.

- Some respond to loss, e.g., TCP Cubic, TCP NewReno.
- Others respond to packet inter-arrival times, e.g., WebRTC, LEDBAT.
- Others respond to per-packet delay, e.g., TCP Vegas.
- Yet others respond to both delay and packet loss, e.g., Compound TCP.

Applications run in diverse network conditions.

- High-speed, low-latency, Data Center networks
- Low-speed, high-latency, transcontinental links
- High-speed, high-variability, bufferbloatd LTE links

Applications have cyclic preferences

Workloads:

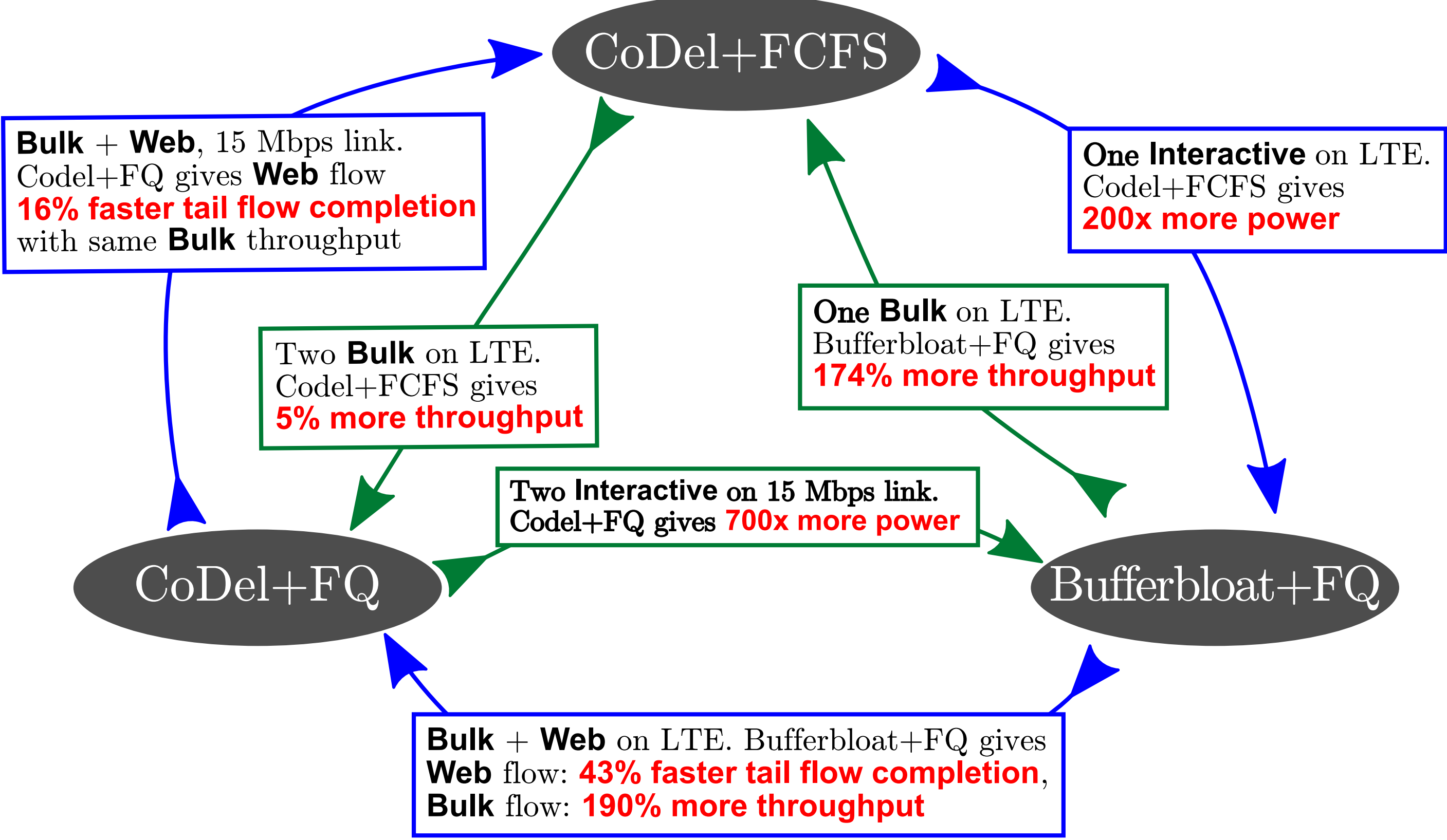
- Bulk:
  - Traffic: A single long-running TCP NewReno stream.
  - Objective: Maximize average throughput.
- Web:
  - Traffic: A single on-off TCP NewReno stream.
  - Objective: Minimize flow completion time at the 99.9th percentile.
- Interactive:
  - Traffic: A single long-running TCP NewReno stream.
  - Objective: Maximize the ratio of average throughput and average round-trip delay (“power”).

In-network schemes:

- A) CoDel+FCFS:
- Queue management: CoDel
  - Scheduling: FCFS
- B) CoDel+FQ:
- Queue management: CoDel
  - Scheduling: Fair Queuing
- C) Bufferbloat+FQ:
- Queue management: Don’t drop any packets.
  - Scheduling: Fair Queuing

Visualizing cyclic preferences

None of the below queue-management and scheduling configurations is best. Power is throughput/delay.  $A \rightarrow B$  indicates that A is better than B.



Key Findings

- Dropping packets on a variable link results in substantial throughput loss.
  - Reason: There is an inherent delay-throughput tradeoff, unlike static links.
- FCFS is preferable to Fair Queuing in some cases.
  - Reason: When competing flows are equally aggressive, they don’t need protection from each other.
- Fair Queuing is required in some cases.
  - Reason: When competing flows are not equally aggressive, they need isolation from each other.
- Dropping packets hurts Flow Completion Time.
  - Reason: Packet drops occur near the end of a flow, preventing DUP ACKs.

The Solution:

Make the data plane more flexible.

- There will never be one conclusive queueing and scheduling scheme.
- Application demands will continue to evolve.
- Networks supporting these application will evolve as well.
- The Data Plane should support new queueing and scheduling schemes.
- Not the same as just picking between multiple existing schemes.

But, do so in a controlled manner.

- Prevent arbitrary programs from running on the switch.
- Allow the network operator to determine only queueing and scheduling.

A blueprint for a flexible data plane

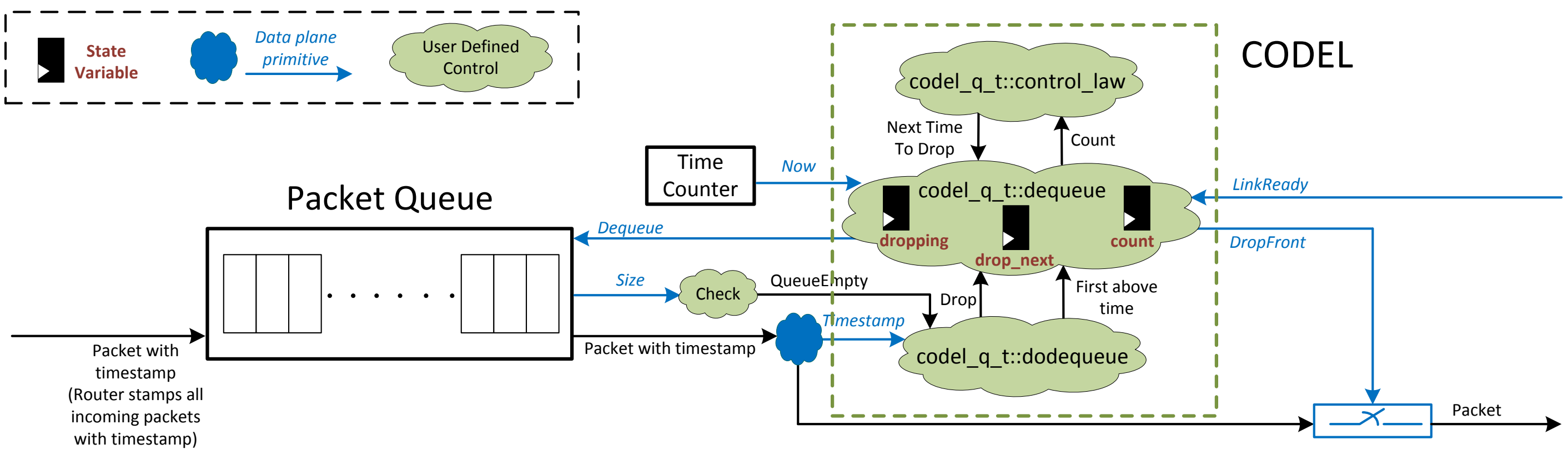
- Standardized interfaces to the rest of the switch

| Class of primitive     | Primitive Name   | Description                        |
|------------------------|------------------|------------------------------------|
| Utilities              | <i>Now</i>       | Get current time.                  |
| Queue primitives       | <i>Size</i>      | Get queue length.                  |
| Queue primitives       | <i>DropFront</i> | Drop packet from head of queue.    |
| Queue primitives       | <i>DropTail</i>  | Drop packet from tail of queue.    |
| Queue primitives       | <i>Enqueue</i>   | Enqueue packet at tail of queue.   |
| Queue primitives       | <i>Dequeue</i>   | Dequeue packet from head of queue. |
| Queue primitives       | <i>Transmit</i>  | Transmit single packet.            |
| Signaling primitives   | <i>LinkReady</i> | Link is ready to accept packet.    |
| Signaling primitives   | <i>Arrival</i>   | New packet just arrived.           |
| Packet primitives      | <i>Timestamp</i> | Packet’s arrival timestamp.        |
| Packet primitives      | <i>Mark</i>      | Set ECN bit.                       |
| Cross-layer primitives | <i>LinkRates</i> | Get current link rate.             |

- Add a small amount of reconfigurable logic to the switch.
- Enforce hard resource limits on the amount of reconfigurable logic.
- Express queueing/scheduling logic as code in a high-level language.

Example implementation: CoDel

- CoDel implementation in SystemVerilog.
- Synthesized into gate-level netlist using Xilinx’s freely available Vivado WebPacket compiler.
- Block diagram of implementation:



- CoDel resource utilization on Xilinx Kintex-7:

| Resource          | Usage                     | Fraction of FPGA |
|-------------------|---------------------------|------------------|
| Slice logic       | 1,257                     | 2%               |
| Slice logic dist. | 1,969                     | 4%               |
| IO/GTX ports      | 27                        | 6%               |
| DSP slices        | 0                         | 0%               |
| Maximum speed     | $12.9 \times 10^6$ pkts/s | N/A              |

Limitations

- Cannot express several important network functions:
  - “Deep-packet” Inspection
  - Intrusion Detection
  - Spam Filtering
- Mechanism for signaling application objectives to the network.
  - Could use DiffServ codepoints.
  - But ASes generally don’t honor codepoints from other ASes.
  - May be a non-issue inside a Data Center.
- Most switches today have a shared pipeline with demanding requirements.
  - Top-of-the-line switches have 64 ports, each running at 10G.
  - Implies that queueing/scheduling logic should run at 64\*10G.
  - If queue management is enqueue-based, e.g., RED, this problem cannot be fixed by replicating digital logic on each port.
- Mechanism to map flows onto per-port queues is required for scheduling.
  - Also need to decide on the number of such queues.
- Interoperability
  - Across switch vendors.
  - Across FPGA vendors.
- Energy and Area overheads.