Dependable End-to-End Delay Constraints for Real-Time Systems using SDN

Rakesh Kumar, Monowar Hasan, Smruti Padhy, Konstantin Evchenko, Lavanya Piramanayagam, Sibin Mohan and Rakesh B. Bobba

The 15th International Workshop on Real-Time Networks
June 27, 2017

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
Oregon State UNIVERSITY OSU

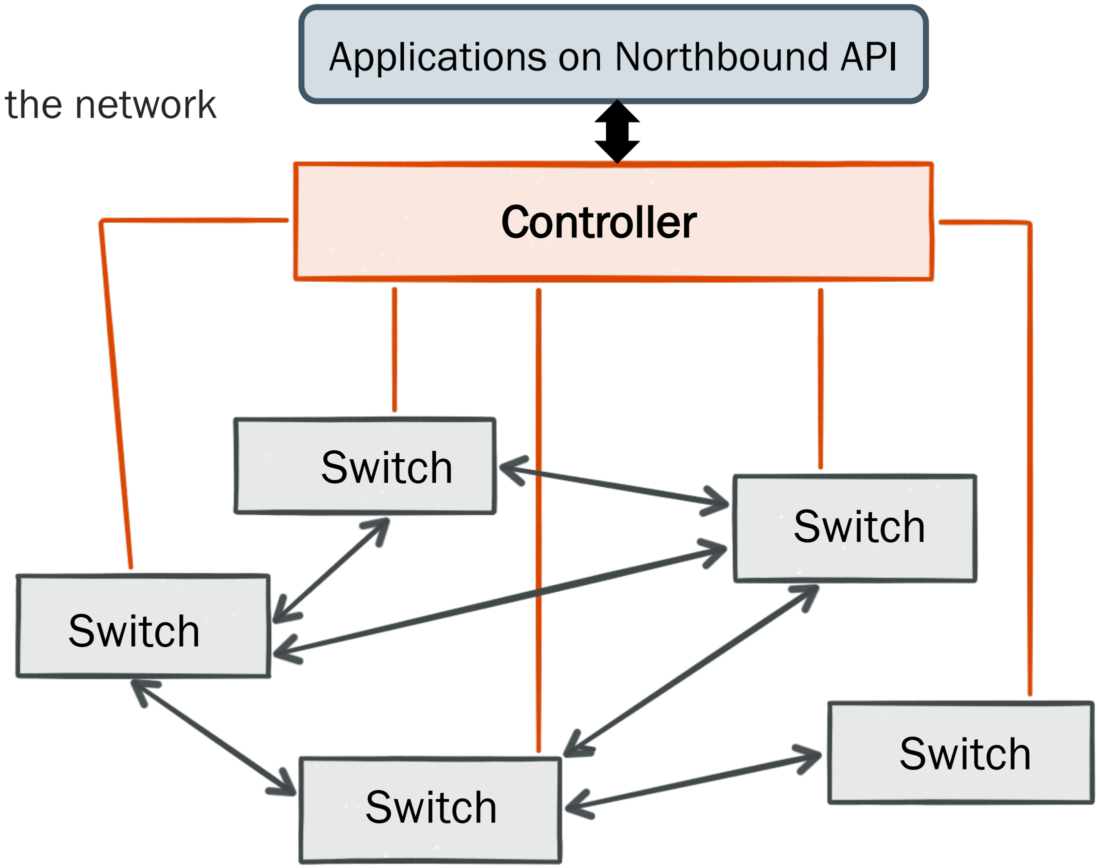
Overview

- Time-critical real-time applications require:
  - A **guaranteed upper bound on the end-to-end packet delay**
  - Avionics, automobiles, industrial control systems, power control networks, etc.

- Current approach: Separate networks for different classes of traffic (high, medium, low criticality)
  - Higher costs
  - Increased management overheads: routers/switches have to be individually programmed
  - Increased attack surfaces

Software Defined Networking (SDN)

- Logically centralized **control plane** at **controller**
- Standardized **data plane** in commoditized **switches** and switch-controller communication protocol
- Controller's **Northbound API**
  - Enables find-grained control of individual flows in the network



SDN Switch

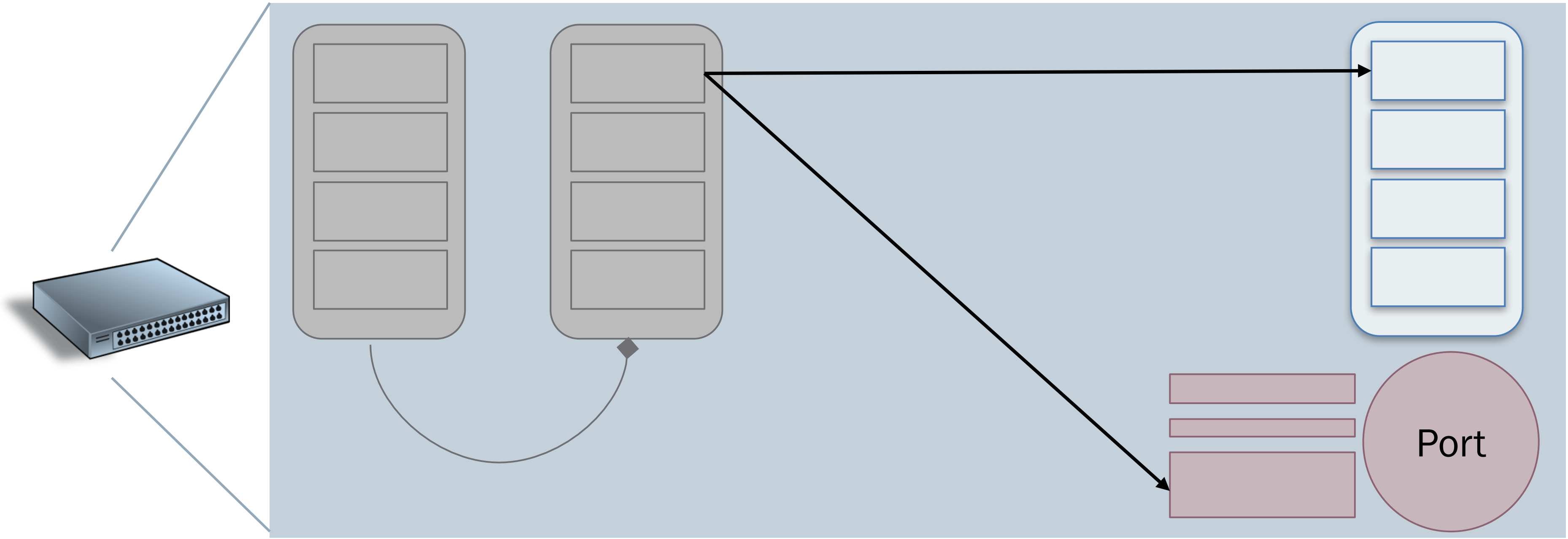
SDN Switch

- Each switch port contains multiple queues

SDN Switch

- Each switch port contains multiple queues
- The entire switch has a meter table

SDN Switch



- Each switch port contains multiple queues
- The entire switch has a meter table
- Flow Tables: Contain matching rules and options to select port, queue and meters

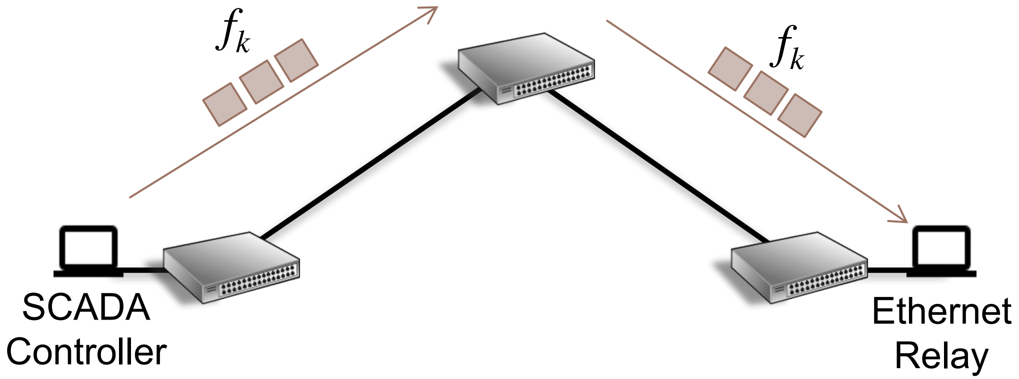
Can SDN Help in Real-Time Systems?

- SDN offers **no end-to-end timing guarantees** for packet flows of individual applications

- SDN and real-time:
  - Can the SDN architecture enable computation of flow paths that meet real-time guarantees?

Problem Overview

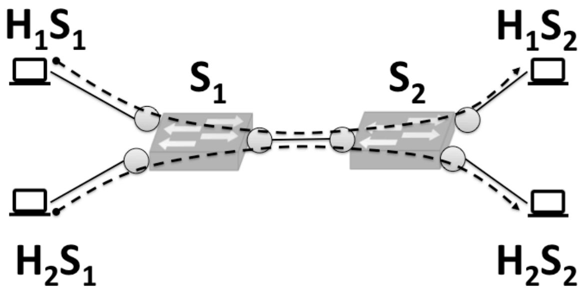
- Each flow (f_k) with **bandwidth** ($\mathrm{B_k}$) and given **end-to-end delay** ($\mathrm{D_k}$) requirements
- **Problem:** allocate n such flows so that the delay and bandwidth constraints are satisfied
  - For **all flows**



Overview/Intuition ➔ Separate Queue for Each High Priority/Critical Flow

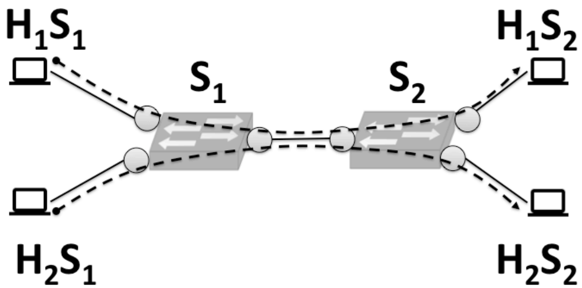
Motivating Example

- Two switch, four host topology
- Two simultaneous flows with different traffic send rates
  - Two different queue configuration:
    1. Each flow has a **separate queue** configured at 50 Mbps
    2. Both flows share **same queue** configured at 100 Mbps

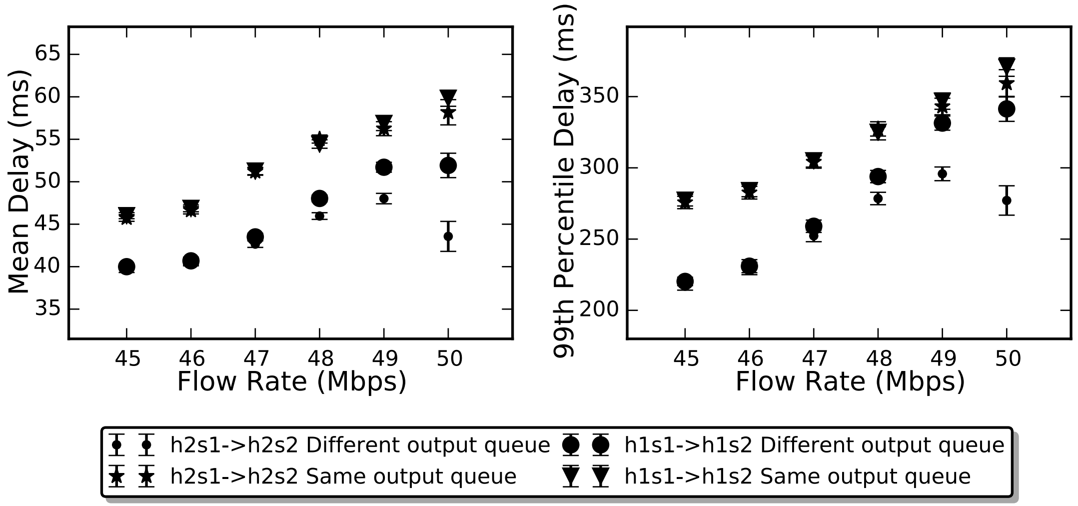


Motivating Example

- Two switch, four host topology
- Two simultaneous flows with different traffic send rates
  - Two different queue configuration:
    1. Each flow has a **separate queue** configured at 50 Mbps
    2. Both flows share **same queue** configured at 100 Mbps



The case with separate queues experiences lower average per-packet delay due to lack of interference



Can SDN Help in Real-Time Systems?

- SDN offers **no end-to-end timing guarantees** for packet flows of individual applications

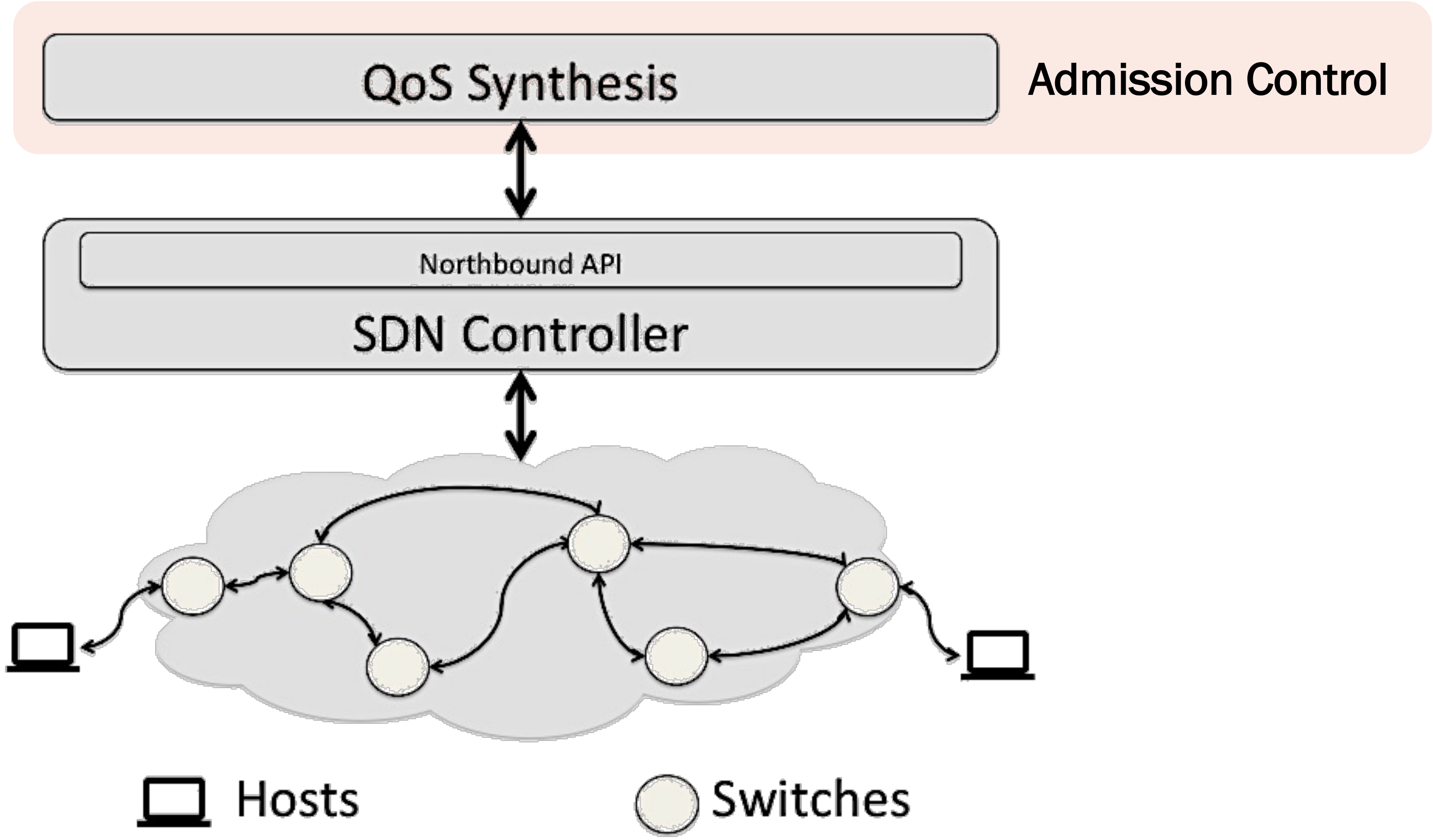
- SDN and real-time:
  - Can the SDN architecture enable computation of flow paths that meet real-time guarantees?

YES

Solution Approach

1. Setup one flow at a time
   - Flows priorities are assigned in **delay-monotonic** order (tighter delay ➔ higher priority)
2. Access system state using the northbound API of the controller
   - E.g.: available resources, network topology
3. Compute the flow path through the SDN such that its requirements are met
   - Solve as a multi-constraint path selection problem
4. Realize path in the SDN topology by using the northbound API

Solution Approach



Solution Approach (contd.)

- End-to-end delay for a given flow can be composed from individual delays at nodes/links:

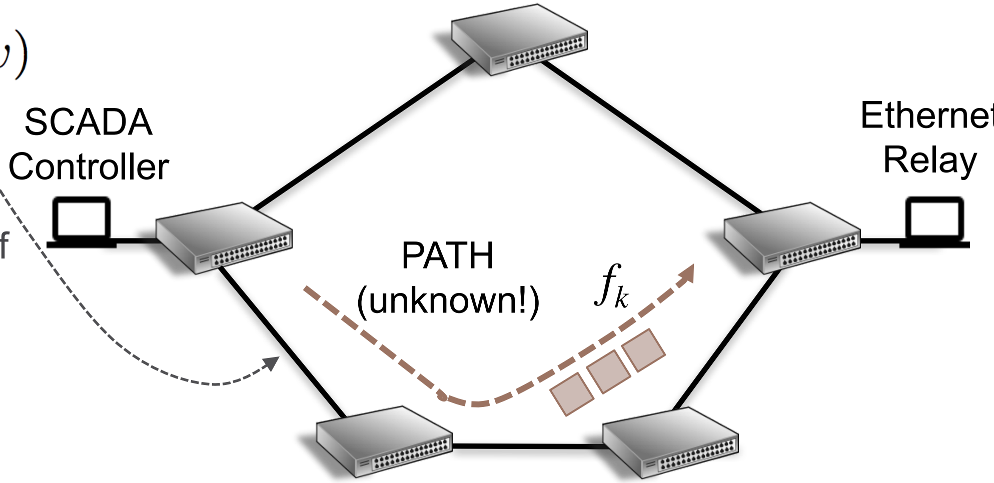
$$\mathfrak{D}_k(\mathcal{P}_k) = \sum_{(u,v)\in\mathcal{P}_k} \mathfrak{D}(u,v)$$

Solution Approach (contd.)

- End-to-end delay for a given flow can be composed from individual delays at nodes/links:

$$\mathfrak{D}_k(\mathcal{P}_k) = \sum_{(u,v)\in\mathcal{P}_k} \mathfrak{D}(u,v)$$

Delay of a link

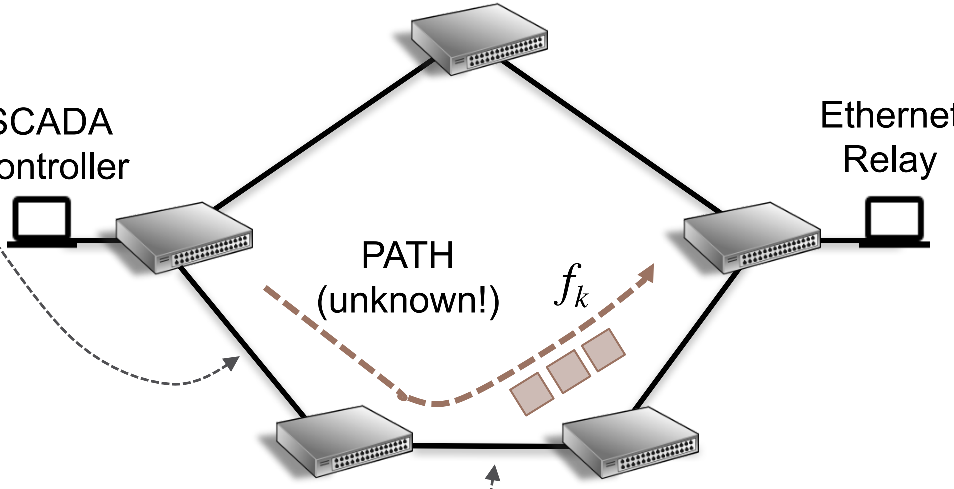


Solution Approach (contd.)

- End-to-end delay for a given flow can be composed from individual delays at nodes/links:

$$\mathfrak{D}_k(\mathcal{P}_k) = \sum_{(u,v)\in\mathcal{P}_k} \mathfrak{D}(u,v)$$

Delay of a link



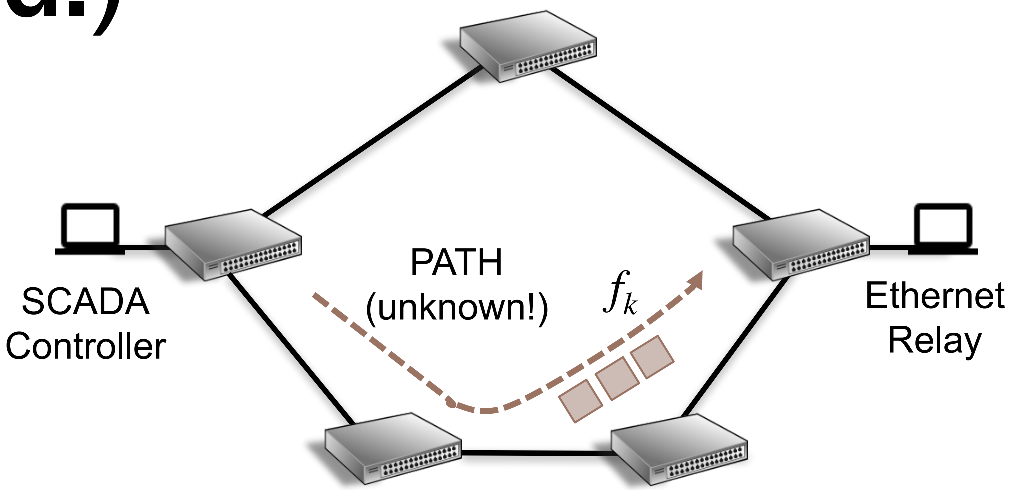
- Bandwidth utilization of the flow on the entire path:

$$\mathfrak{B}_k(\mathcal{P}_k) = \sum_{(u,v)\in\mathcal{P}_k} \mathfrak{B}_k(u,v)$$

Required bandwidth utilization of a link

Solution Approach (contd.)

Multi-Constraint Path (MCP) Selection



- Delay constraint
  - Total delay over path less than end-to-end delay budget

$$\mathfrak{D}_k(\mathcal{P}_k) \leq D_k$$

- Bandwidth constraint
  - Flow bandwidth utilization on all links can fit within the total utilization along the path

$$\mathfrak{B}_k(\mathcal{P}_k) \leq \max_{(u,v)\in E} \mathfrak{B}_k(u,v)|V|$$

- Shortest-path may **NOT** satisfy both the constraints!
  - MCP is NP-Complete!
  - Developed a polynomial heuristic to solve this multi-constraint problem ➔ calculate paths

Solution Approach (Contd.)
Path Realization Using Intents

- **Intent ➔** actions performed on the packets in a given flow at an individual switch
- Each intent is 4-tuple given by

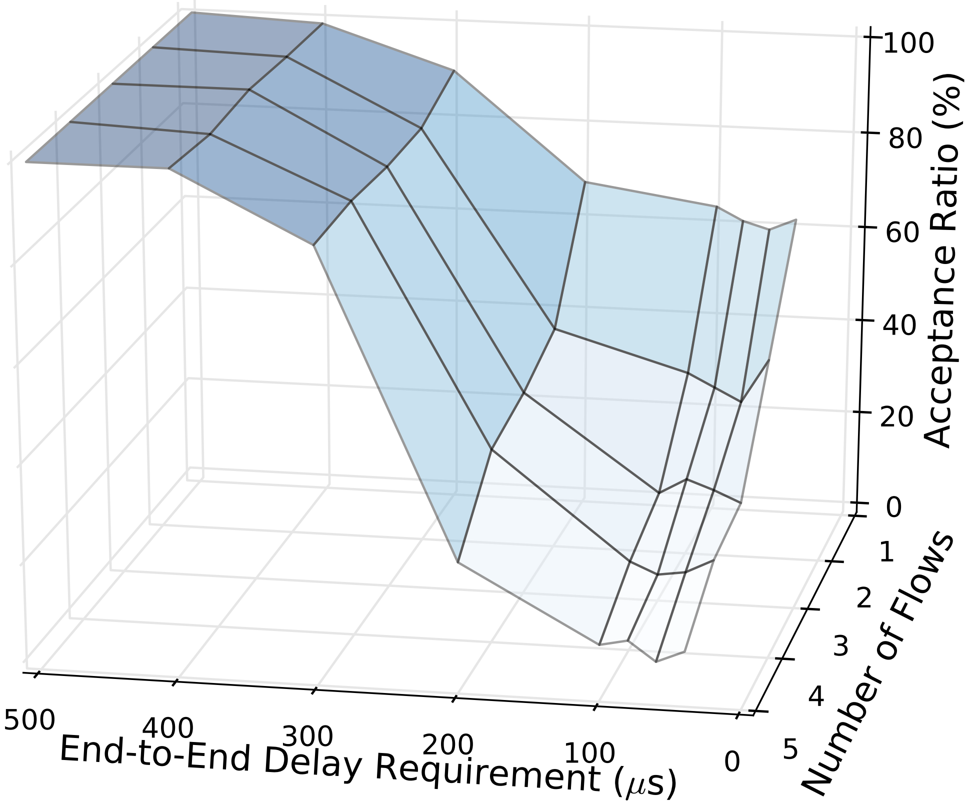
$$(\mathsf{Match}, \mathsf{InputPort}, \mathsf{OutputPort}, \mathsf{Rate})$$

- Intents are realized with a **flow rule** and a corresponding **exclusive queues**

Evaluation
Setup

- Experiments performed on a machine running **Mininet** and **RYU**
  - Python implementation of northbound application for QoS Synthesis
- **250** random topologies: five switches, each switch having two hosts
- Each link has the bandwidth of **10 Mbps**
- Link delays: generated uniformly randomly between [**25, 125**] microseconds
- Bandwidth requirements: randomly generated between [**1, 5**] Mbps
- [**1, 5**] real-time and [**1,3**] non-real time flows are generated using **Netperf**
  - Each flow lasts for 10 seconds

Results

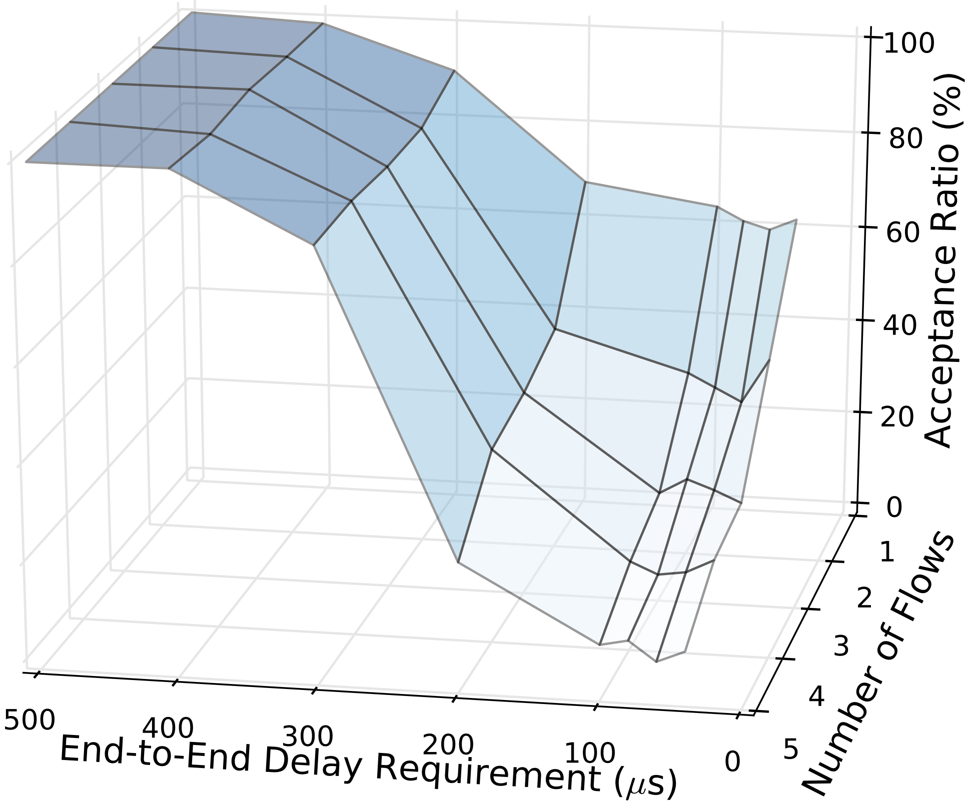


X-axis: Delay requirements

Y-axis: Number of flows

Z-axis: % of *schedulable* flows

Results



X-axis: Delay requirements

Y-axis: Number of flows

Z-axis: % of *schedulable* flows

The acceptance ratio **decreases** with:

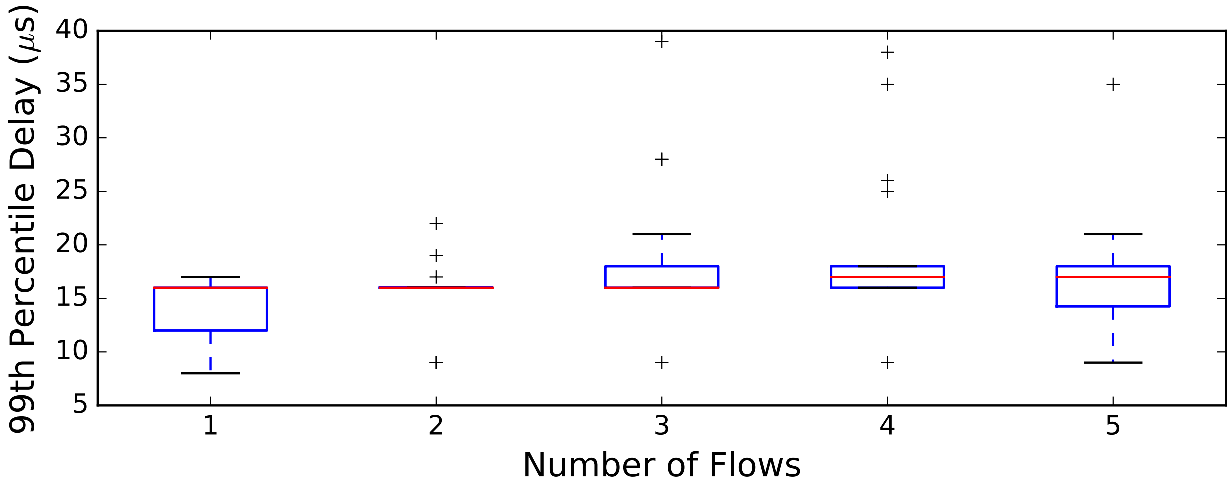
1. Increasing the number of flows; or
2. For stringent end-to-end delay requirements

ILLINOIS UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN Oregon State UNIVERSITY OSU

Results

X-axis: Number of flows

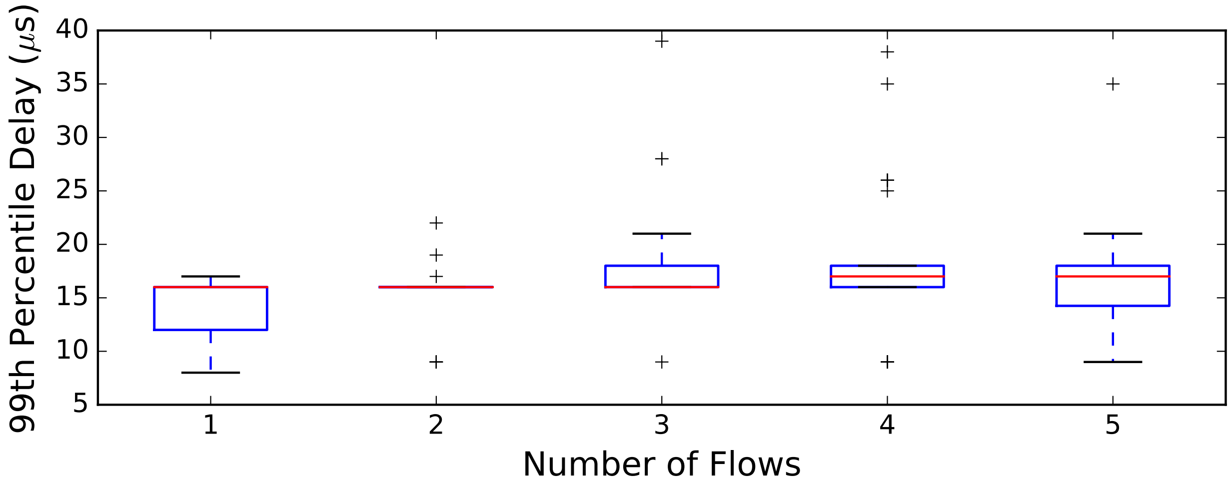
Y-axis: Observed delay (99th percentile)



Results

X-axis: Number of flows

Y-axis: Observed delay (99th percentile)



1. Non-real time flows **do not cause** interference for real-time flows
2. Increasing the number of real-time flows increases end-to-end delay

Conclusion

- Our approach:
  - Successfully allocate flows for highly critical RTS network traffic on SDN architectures
  - Non-critical flows do not interfere with critical ones
  - Useful for COTS systems
- The evaluation results are another instance of the “No Free Lunch Theorem”
  - The acceptance ratio decreases either
    - With increasing the number of flows or
    - Stringent end-to-end delay requirements
- Open Issues
  - What does the optimal allocation look like?
  - Multiplexing the usage of a single queue for multiple flows remains an open problem

RTSS Prepint
https://arxiv.org/abs/1703.01641

Thank You!

Questions?