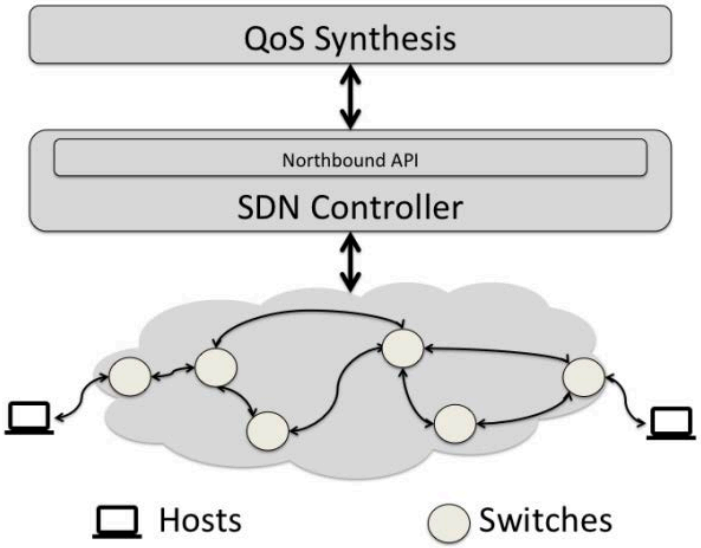
End-to-End Delay Guarantees for Real-Time Systems using SDN

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

Motivation

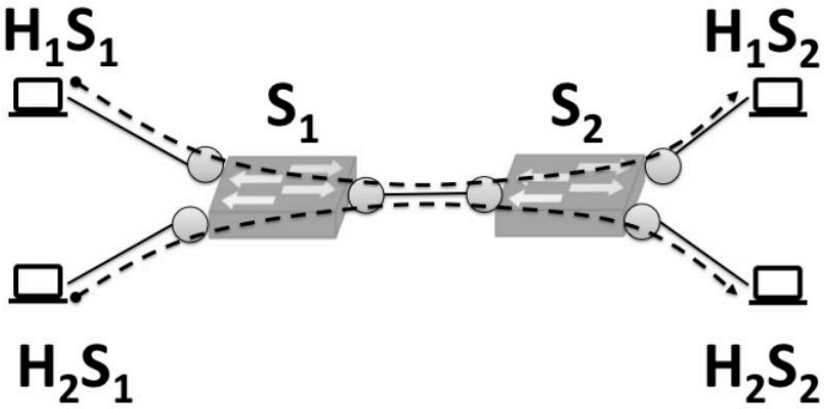
- Real-time systems (RTS) require that timing critical applications' packets from one host to another are delivered with a guaranteed upper bound on the *end-to-end* packet delay.
  - e.g. smart grids, avionics, automobiles, industrial control systems
- Current approach: Separate networks for different classes of networks:
  - Higher costs and management overheads
  - Increased attack surface

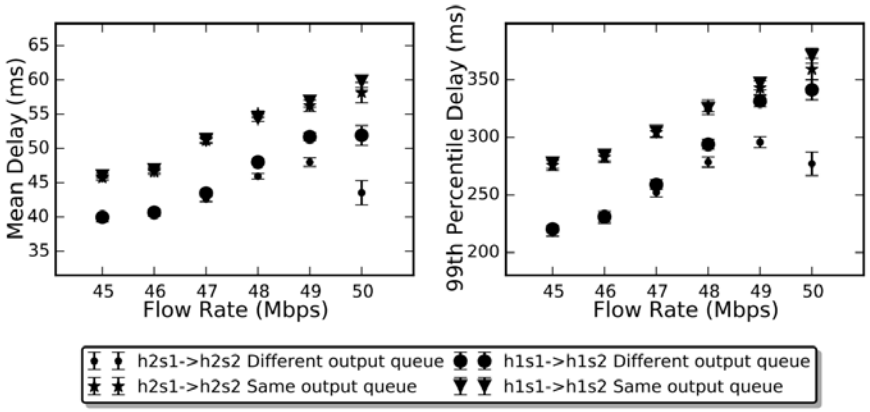
Software Defined Networking (SDN)



- 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

Motivating Example





- Two simultaneous flows with traffic at varying send rates. Two cases for queue configuration:
  - Each flow has a separate queue configured at 50 Mbps.
  - Both flows share a queue configured at 100 Mbps
- The case with separate queues experiences lower average per-packet delay due to lack of interference.

Can the SDN Architecture Help?

- The architecture offers no QoS guarantees for individual applications' packet flow paths.

- Questions:
  - Can the SDN architecture enable computation of flow paths that meet the QoS guaranteed specified by the network operators?

    Yes!

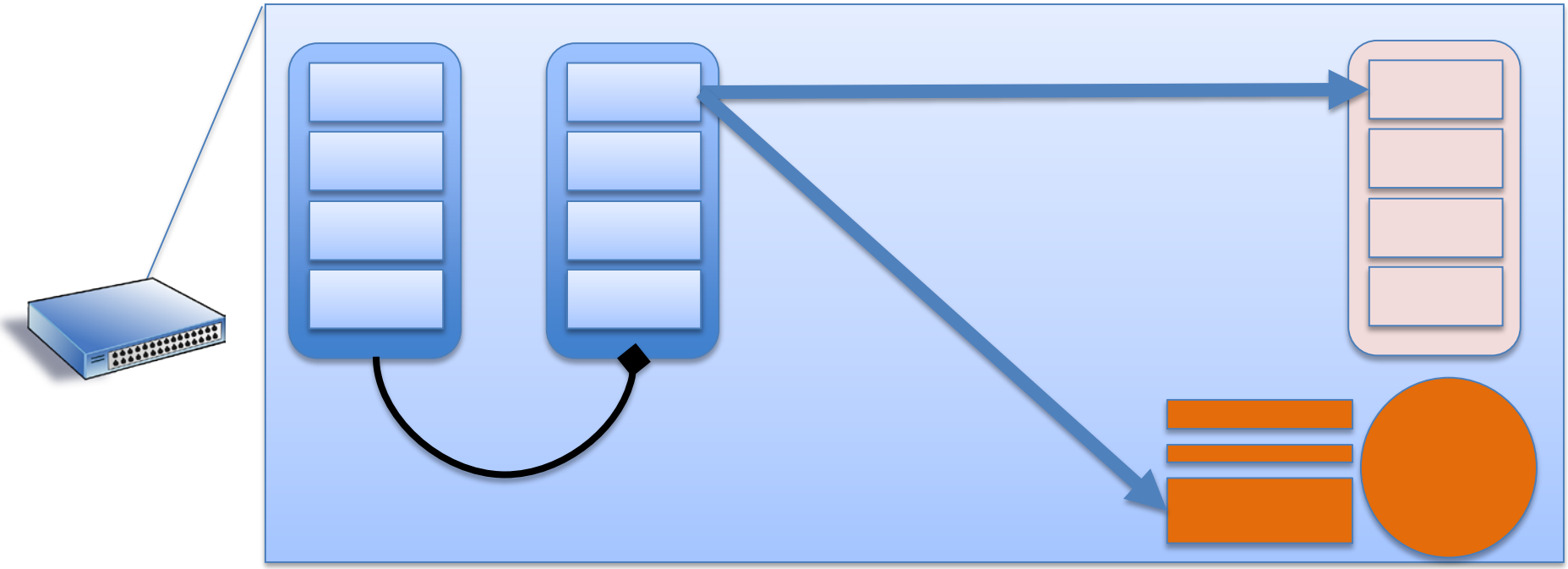
  - Can the SDN architecture be used to allocate resources for individual network flows?

    Sure...

Rest of the talk

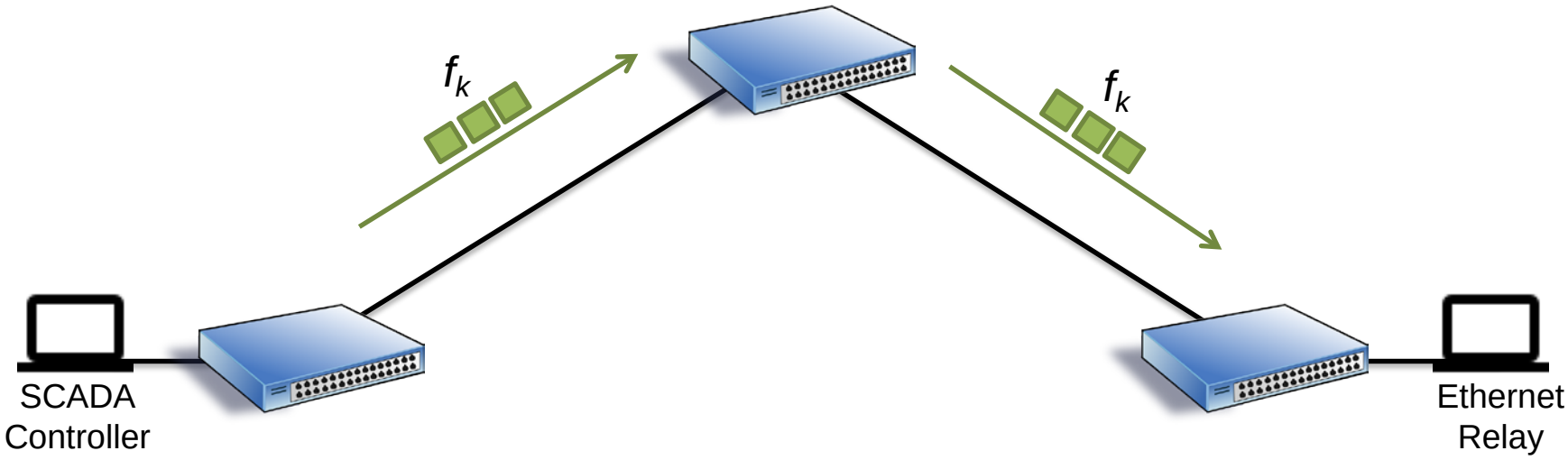
- Life of a packet in an SDN switch
- Problem and Solution Overview
- System Model
- Multi-Constrained Path Problem
- Evaluation
- Conclusion and Future Work

Life of a Packet in an SDN switch



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

Problem Statement



- Each flow (f_k) with bandwidth and delay requirements given by B_k and D_k.
- Allocation of n such flows so that their bandwidth and delay requirements are satisfied.

Solution Overview

- Setup one flow at a time, starting with the flow with tightest delay requirements.
- Access the system state (i.e. available resources, network topology) using the northbound API of the controller.
- Finally:
  - Compute the flow path through the SDN such that its requirements are met.
  - Realization of path in the SDN by again using the northbound API.

System Model - I

- Consider a graph (*V, E*) where:
  - Nodes (*V*) are all the ports in the network.
  - The edges (*E*) are come from:
    - Topology
    - If two ports are on the same switch, they are connected.

System Model - II

- The total delay for a given path can be composed for the end-to-end path delay:

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

- The total bandwidth consumed by the flow on the entire path is given by:

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

Multi-Constrained Path Problem

- Delay Constraint:

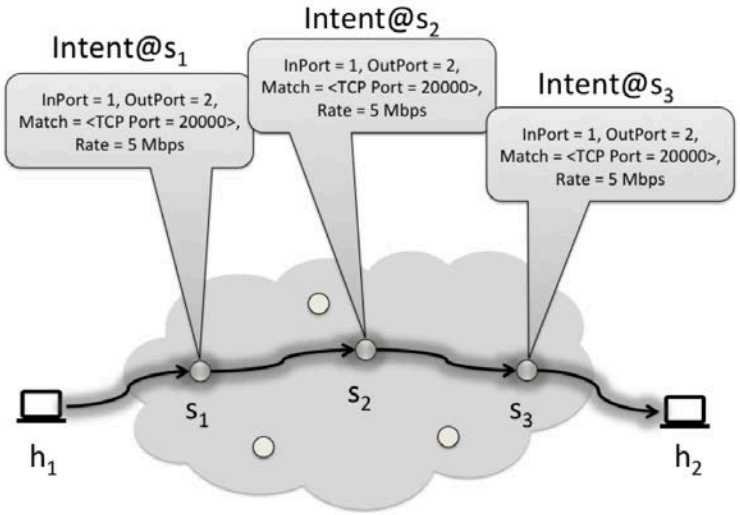
$$\mathfrak{D}_k(\mathcal{P}_k) \leq D_k.$$

- Bandwidth Constraint:

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

- NP-Complete but polynomial time heuristic available.

Path Realization



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

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

- Intents are realized with a flow rule and a corresponding exclusive queue.

Evaluation - Setup

TABLE I
EXPERIMENTAL PLATFORM AND PARAMETERS

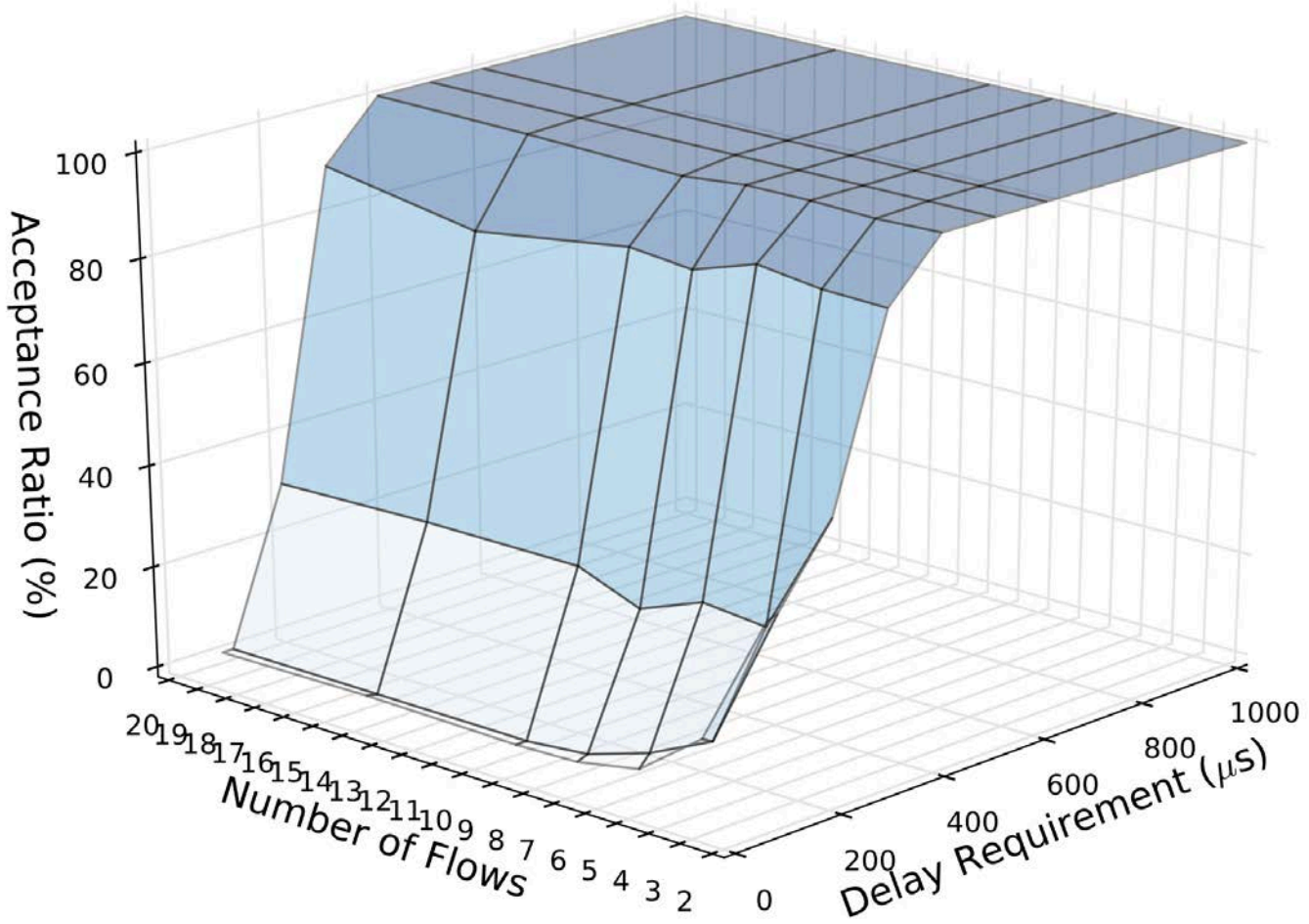
Artifact/Parameter	Values
Number of switches	5
Bandwidth of links	10 Mbps
Bandwidth requirement of a flow	[1, 5] Mbps
SDN controller	Ryu 4.7
Switch configuration	Open vSwitch 2.3.0
Network topology	Synthetic/Mininet 2.2.1
OS	Debian, kernel 3.13.0-100

Randomly generated topologies by adding random links to a ring.

Evaluation: How many flows can be packed?

- Random link delays between [25, 125] us.
- For each flow, pick:
  - D_k is a function of the randomly generated topologies.
    - Let D_{min} = [200, 1000] us be the lowest delay for a flow.
    - Increment by $D_{min}/10$ for each other flow.
- For each choice of delay requirement and number of required flows, generated 250 random instances.
  - The acceptance ratio is the instances that successfully admitted all the required flows.

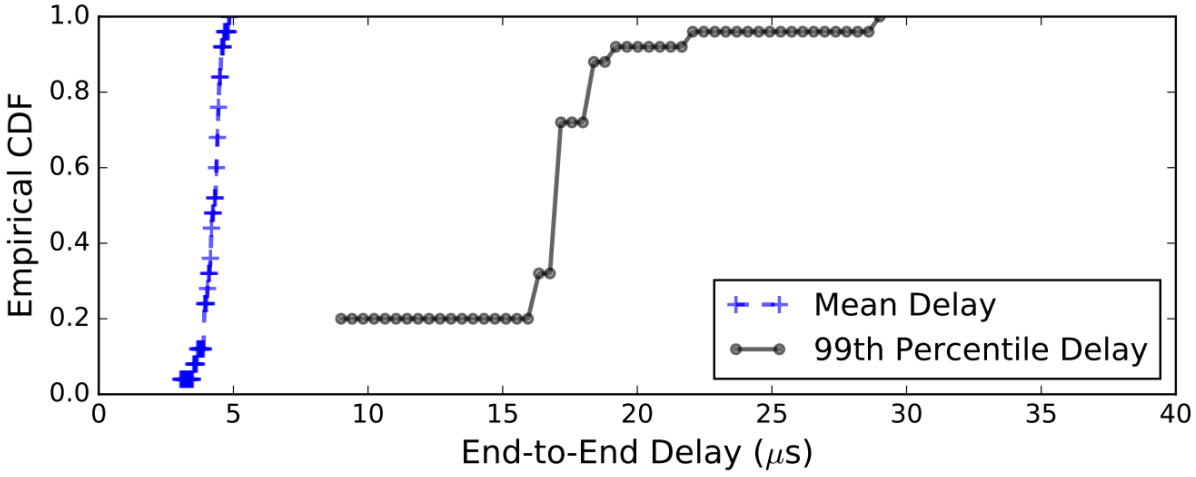
Evaluation: How many flows can be packed?



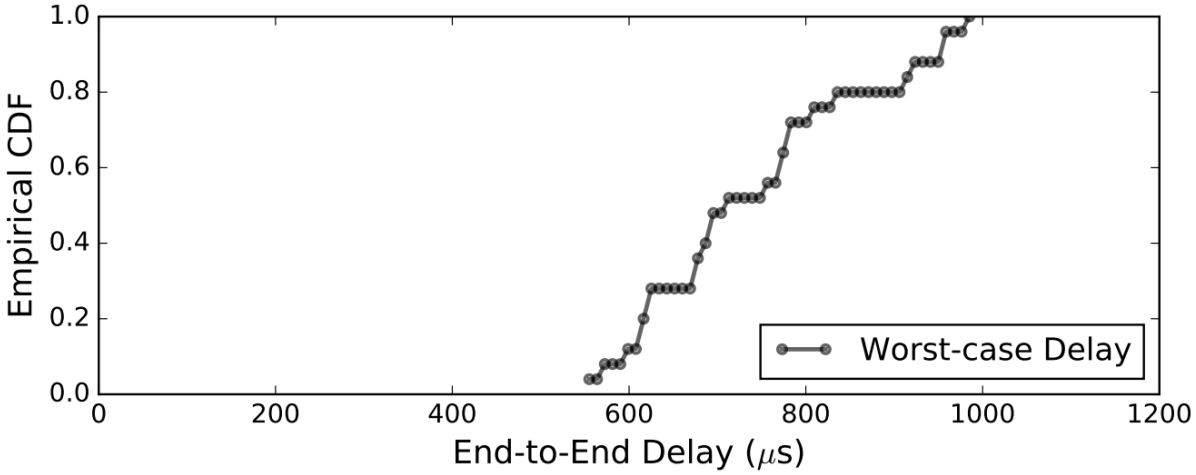
Evaluation: Can the flows be realized?

- Link delay set to zero.
- Added [1, 3] non-critical background flows.
- Seven critical flows.
- Each flow is CBR UDP traffic generated using `netperf` which lasts for 10 seconds:
  - D_k:
    - D_{min}: 100 us * diameter of the topology (i.e. ~4).
    - For others, increment by 10 us for each flow.

Evaluation: Can the flows be realized?



(a)



(b)

Conclusion and Discussion

- COTS successfully used to allocate flows for highly critical RTS network traffic by exploiting opportunities presented by SDN.
  - Multiplexing the usage of a single queue by multiple flows remains an open problem.
- 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.
  - What does the optimal allocation look like?