NSF CC*: Q-Factor

Real-time transfer optimization using programmable data planes

Jeronimo Bezerra, FIU, PI

Dr. Richard Cziva, ESnet, Co-PI

Dr. Julio Ibarra, FIU, Co-PI

NSF Program Director: Dr. Deep Medhi

Outline

Motivation [1/4]

Host tuning is necessary for any new OS installation

Complex and time-consuming (HW, OS, SW, RTT, TCP vs. UDP)

Maxing out tuning parameters waste host’s resources

No single tuning configuration fits all configurations of HW and SW

Over time, context might change (new HW, higher/lower RTT)

What if tuning could be done dynamically based on HW, OS, SW, RTT, and transport?

Motivation [2/4] – The importance of proper tuning

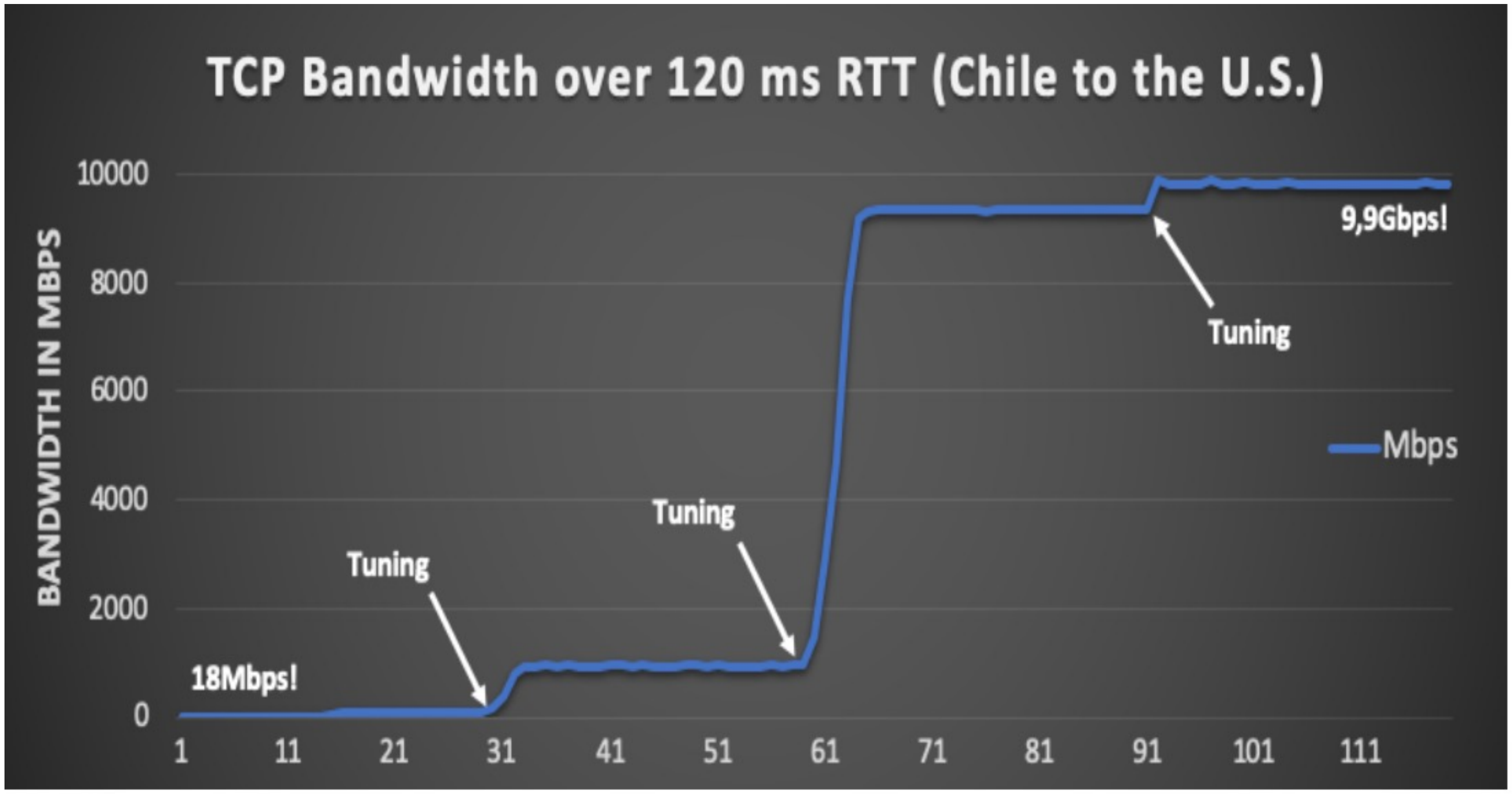
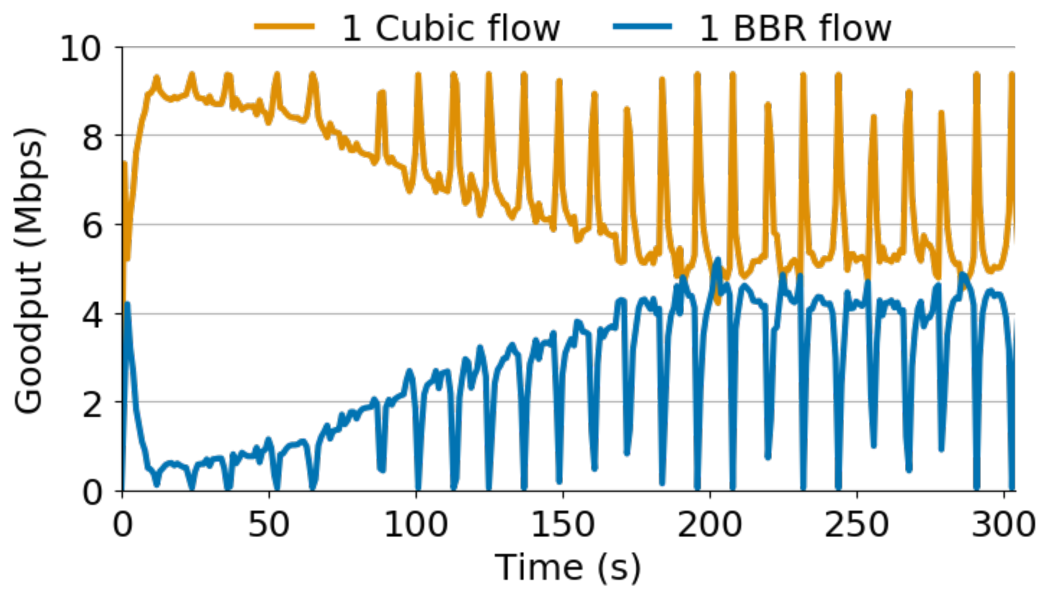


Figure 1 TCP Bandwidth over 120ms RTT when tuning is applied

Motivation [3/4] - TCP Congestion Control 101

- Ideally TCP would send packets at a fair share of network capacity. But the TCP sender has no idea what "available network capacity" means
- As a result, TCP algorithms probe the network by slowly increasing their sending rate
- Usual indication of a "that's too fast" condition is packet drops (no ACK arrives)
- This is a key behavior in different congestion control algorithms
- In fact, TCP is an ACK Pacing protocol that only infers congestion



- *A packet is dropped when a forwarding device's buffer/queue is full or close to full.*

Motivation [4/4] - How to detect if queues are forming?

- Several possible solutions:
  - ICMP Source Quench Redux
    - ICMP generates a congestion notification (and tells which TCP flow)
    - Unreliable (often filtered out) and reactive
  - Explicit Congestion Notification
    - Sparse signal (single bit)
    - Both hosts need to be aware, IP-level marking needs modification at the end hosts
  - Use newer (i.e., delay-based) congestion control (BBR 2.0)
  - …
  - <u>Still, congestion control only infers network state</u>
  - *What if the network device/router/switch could share the info?*

Q-Factor's Goals

Offload tuning from IT operators

- Telemetry agents on the hosts doing the tuning or making suggestions to operators

Dynamically tune hosts based on

- Hardware capabilities
- OS configuration
- +RTT
- +network conditions

Leveraging new telemetry technologies

- P4/In-band Network Telemetry
- Programmable NICs
- Linux eBPF & XDP

Why now? What has enabled Q-Factor?

Challenge:

- Queue occupancy leads to packet drops and jitter, and it varies per packet received
- Currently, host protocols *infer* queue occupancy
- Network devices export queue occupancy values in summaries

What has changed?

- Development of the P4 In-band Network Telemetry (INT) application in 2016
- Launch of the Barefoot/Intel Tofino ASIC in 2016
- Launch of the first programmable NICs in 2016/2017
- Launch of the first Tofino white boxes in 2018

Leveraging In-band Network Telemetry [1/3]

INT is a P4 application that records network telemetry information in the packet while the packet traverses a path between two points in the network

As telemetry is exported directly from the Data Plane, the Control Plane is not affected:

Translating: *you can track/monitor/evaluate EVERY single packet at line rate and real time.*

Examples of telemetry information added:

Timestamp, ingress port, egress port, queue buffer utilization, sequence #, and many others

Leveraging In-band Network Telemetry [2/3]

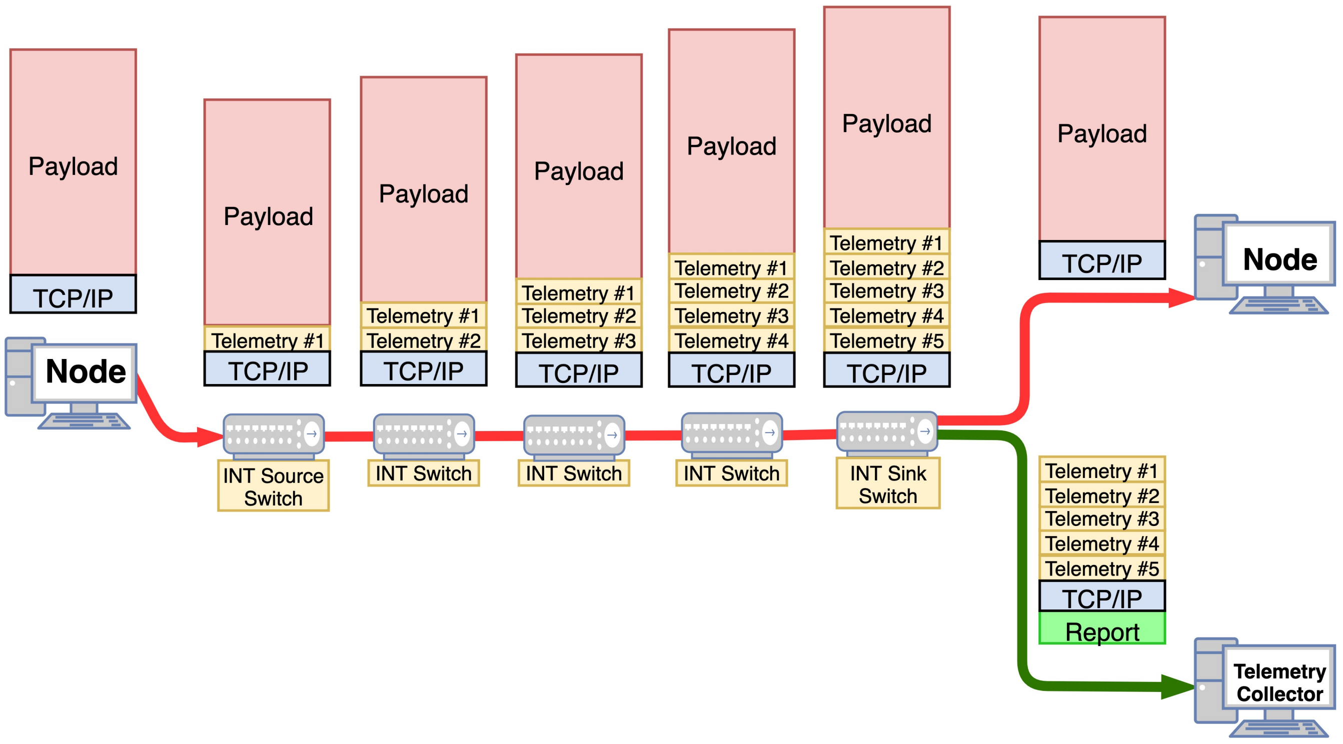
1 – User sends a TCP or UDP packet unaware of INT

2 – First switch (INT Source Switch) pushes an INT header + metadata

3 – Every INT switch pushes its metadata. Non-INT switches just ignore INT content

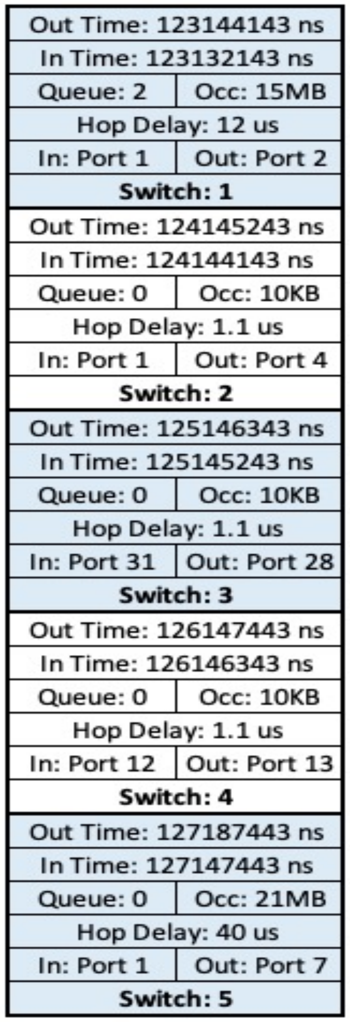
4 – Last switch (INT Sink Switch) extracts the telemetry and forwards original packet to destination

5 – Last switch (INT Sink Switch) forwards the 1:1 telemetry report to the Telemetry Collector



Leveraging In-band Network Telemetry [3/3]

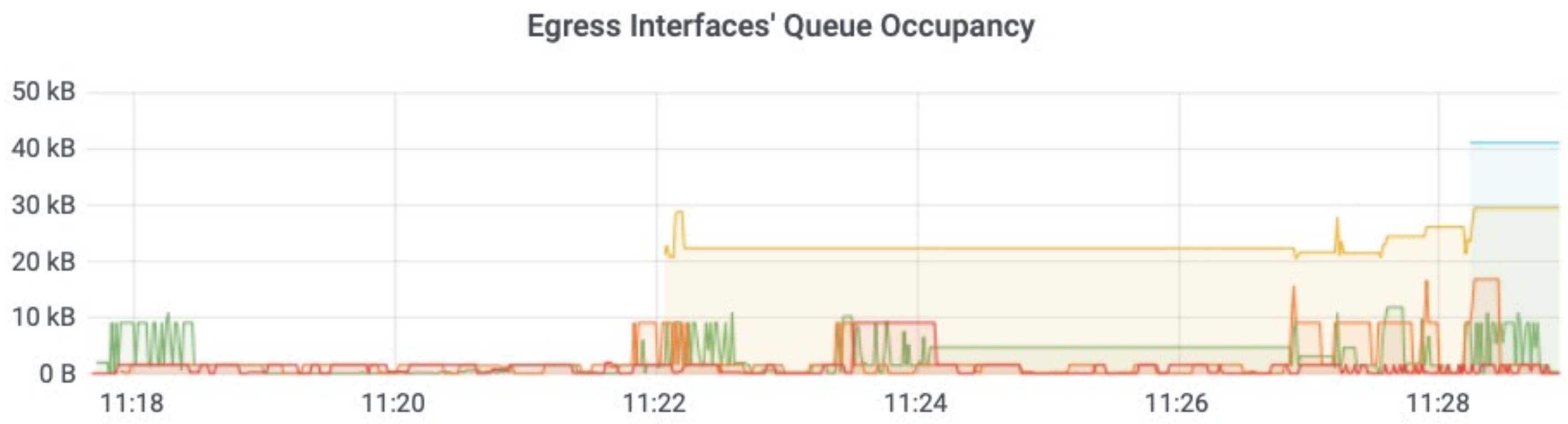
- The telemetry data supported by our P4/Tofino switches:
  - Per switch:
    - Switch ID
    - Ingress port
    - Egress port
    - Ingress timestamp
    - Egress timestamp
    - Egress queue ID
    - Egress queue occupancy
  - Per telemetry report:
    - Report timestamp
    - Report sequence number
    - Original TCP/IP headers



Out Time: 123144143 ns	
In Time: 123132143 ns	
Queue: 2	Occ: 15MB
Hop Delay: 12 us	
In: Port 1	Out: Port 2
Switch: 1	
Out Time: 124145243 ns	
In Time: 124144143 ns	
Queue: 0	Occ: 10KB
Hop Delay: 1.1 us	
In: Port 1	Out: Port 4
Switch: 2	
Out Time: 125146343 ns	
In Time: 125145243 ns	
Queue: 0	Occ: 10KB
Hop Delay: 1.1 us	
In: Port 31	Out: Port 28
Switch: 3	
Out Time: 126147443 ns	
In Time: 126146343 ns	
Queue: 0	Occ: 10KB
Hop Delay: 1.1 us	
In: Port 12	Out: Port 13
Switch: 4	
Out Time: 127187443 ns	
In Time: 127147443 ns	
Queue: 0	Occ: 21MB
Hop Delay: 40 us	
In: Port 1	Out: Port 7
Switch: 5	

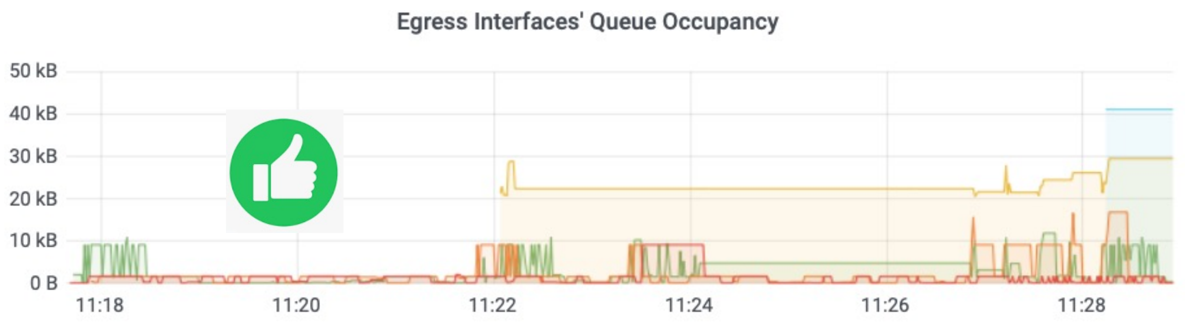
Leveraging INT – Mitigating Buffer/Queue Utilization [1/2]

- Instantaneous Egress Interface Queue utilization (or buffer)
  - Metadata needed:
    - List of switches and their metadata
    - Per switch:
      - Switch ID, Egress port, Egress Queue ID, Queue Occupancy

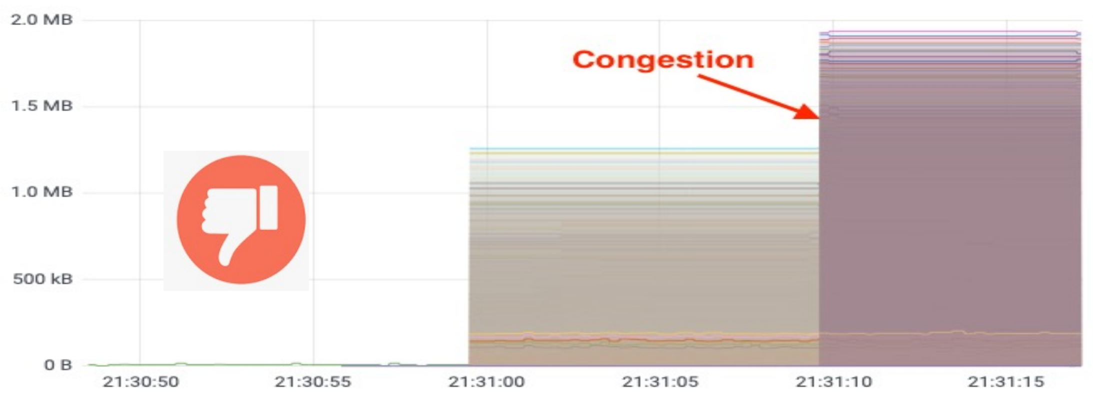


Leveraging INT – Mitigating Buffer/Queue Utilization [2/2]

- Ideal Buffering:
  - 1-10's KBs of queueing
  - Not noticeable by users or applications

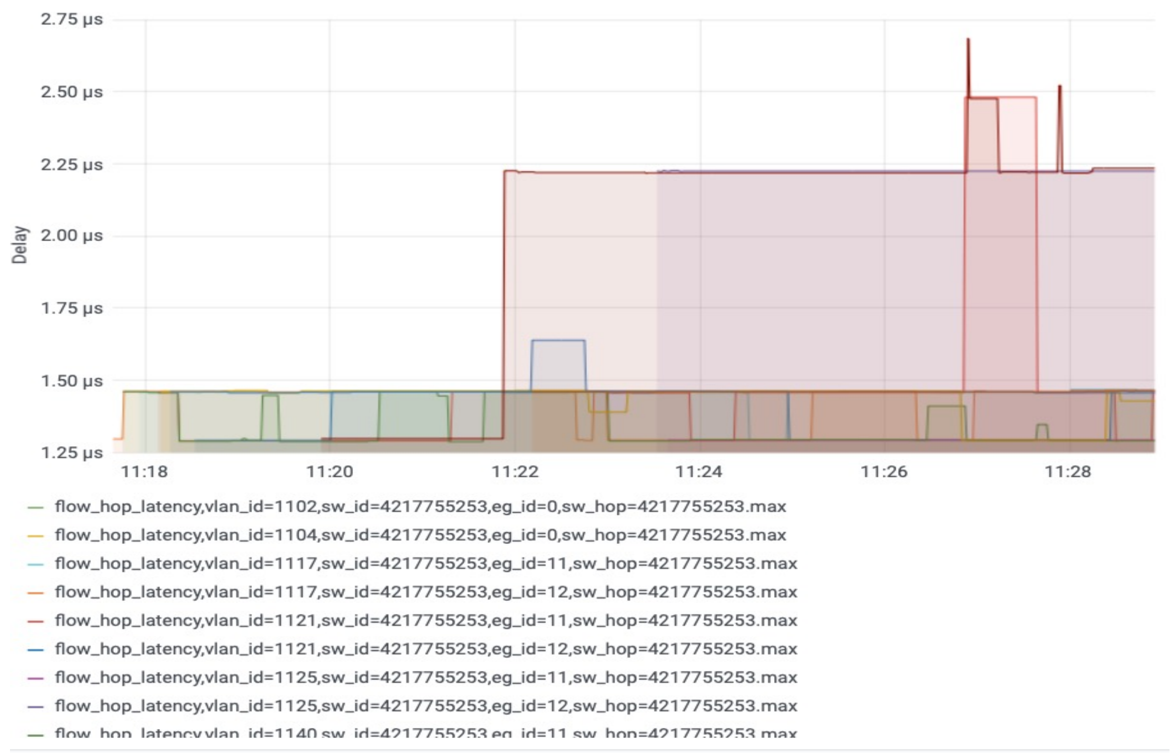


- Congestion:
  - 1000-10000's KBs of queueing
  - Dramatically noticeable by users or applications via increased delay and packet drops (retransmissions).



Leveraging INT – Observing Hop Delay & Jitter [1/2]

- Metadata needed:
  - List of switches
  - Per switch:
    - Switch ID
    - ingress timestamp
    - egress timestamp

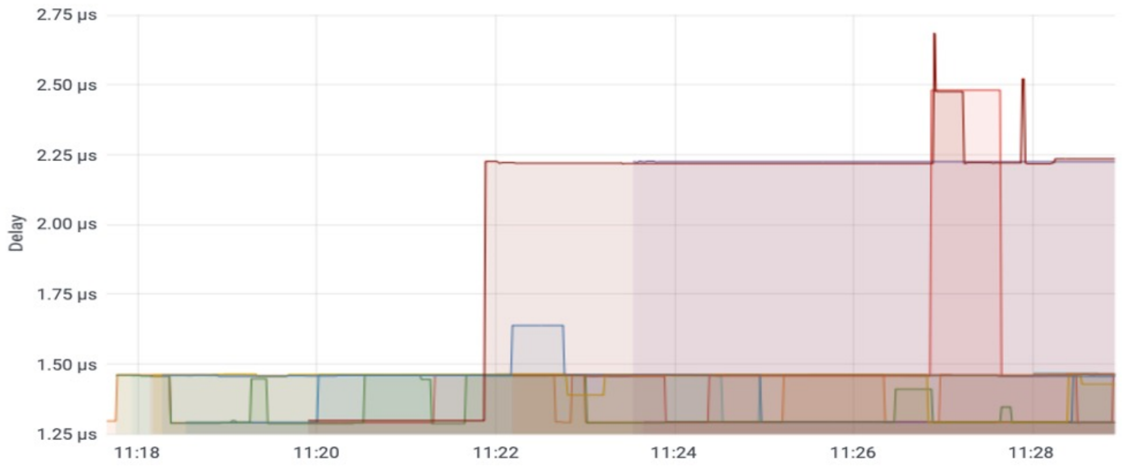


Leveraging INT – Observing Hop Delay & Jitter [2/2]

Sources of jitter

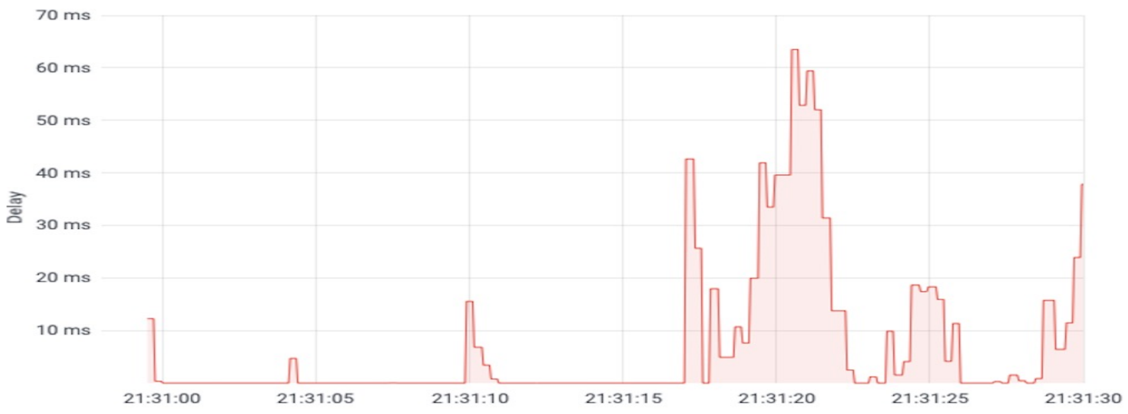
Ideal Hop Delay:

- 1-10’s Microsecond's delay
- Barely noticeable by users or applications in 10+ms RTT environments



Hop Delay during a burst:

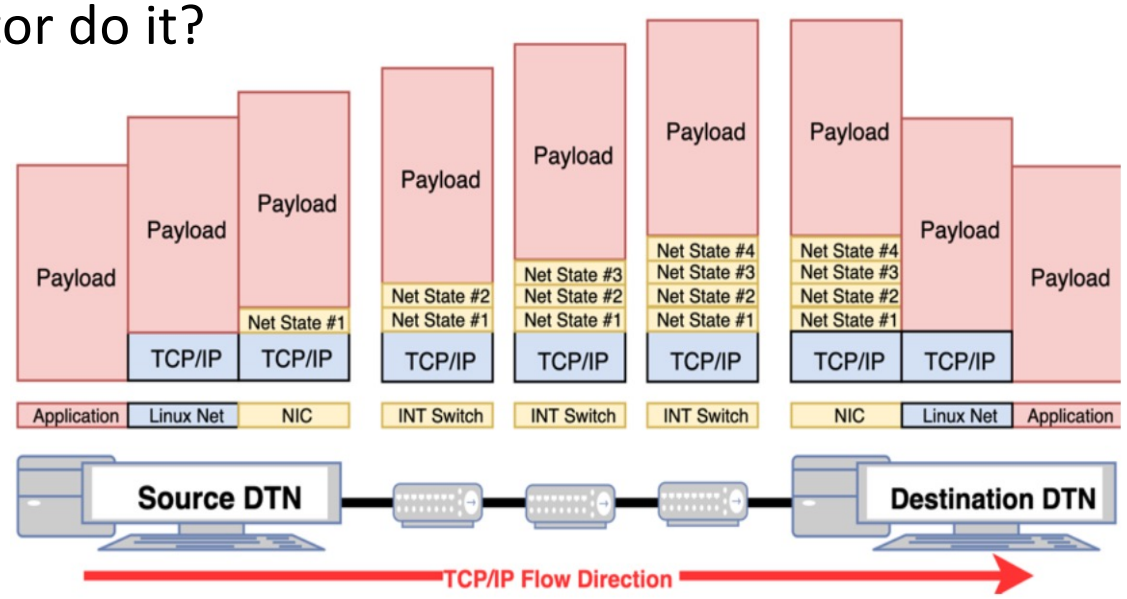
- 1000-10000’s Microsecond's delay
- Dramatically noticeable by users or applications for all environments



- P4/INT enables network operators to have even deeper visibility
  - Better QoS policies
  - Easier-to-verify tuning configuration
  - Fast mitigation of sources of jitter and packet drops
- However, P4/INT focus ONLY on network operators
  - Users remain unaware of what is happening when their applications “struggle”
  - Users *over* tune their hosts based on the hosts’ resources available or expectations
  - Users still have to infer the network state

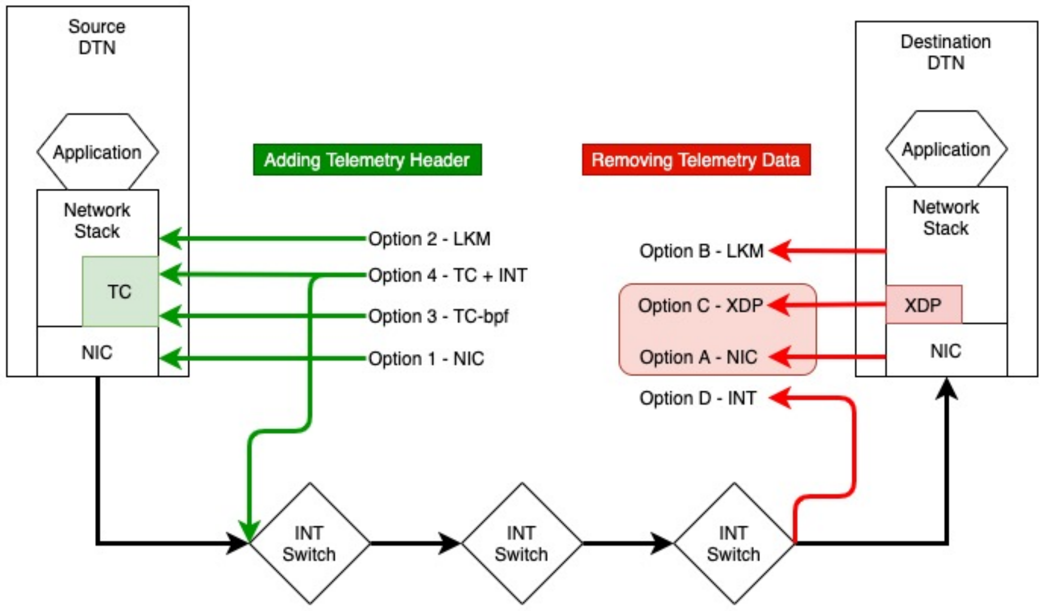
Deploying a Distributed Management Plane

- What if we extended the Management Plane to the hosts?
- What if hosts could access telemetry metadata and find out the network state instead of just inferring it? Would that be useful?
- How will Q-Factor do it?



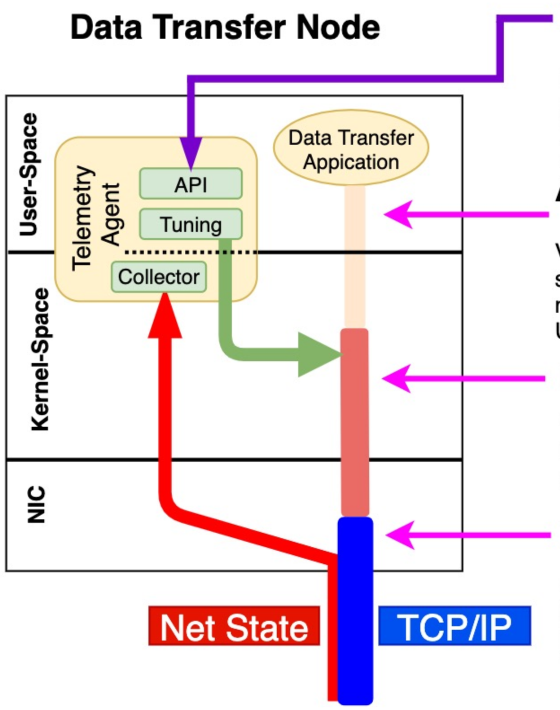
How will Q-Factor operate? [1/2]

- Q-Factor will extend the hosts' capabilities by:
    - Adding support for INT to hosts
    - Supporting a Telemetry Agent to tune the host
- INT at hosts will operate via two approaches:
    - If a programmable NIC is available, INT will be done at the NIC
    - Otherwise, eBPF/XDP/TC will be used before the Linux TCP/IP stack for performance
- Hosts' applications will NOT need to be changed:
    - Q-Factor will operate before the Linux TCP/IP stack
    - Completely hidden from the upper layers



How will Q-Factor operate? [2/2]

- Q-Factor's Telemetry Agent will tune the hosts:
  - Hosts will be pre-tuned based on best practices (buffers, BIOS, NIC, MTU, etc.)
  - Hosts will be dynamically tuned based on INT metadata
- Operators will have control over tuning:
  - *Learning* mode just with tuning recommendations
  - *Active* mode dynamically applying the tuning recommendations



External API

Using gRPC, Telemetry Agent receives tuning instructions and instructions/policies from network operators.

Application-Layer

Via netlink/sockets, data (payload) is sent down to Linux kernel. Concurrency might be used for performance. Unaware of any optimization.

Kernel-Space

Telemetry Agent sets MSS/buffering/tuning using real-time network state data.

Physical-Layer/NIC

Adds a network state header to outgoing traffic and extracts and forwards it to the Telemetry Agent for incoming traffic.

Building the Q-Factor Telemetry Agent

- Q-Factor's major contribution:
  - Consume telemetry metadata to tune hosts
  - Tune hosts based on best practices and hardware configuration
  - Save memory by setting buffers to the proper size by using the active data transfers' highest RTT
  - Processing INT's queue occupancy and hop delay to mitigate bottlenecks
  - *Even if you don't have INT in your network, Q-Factor will offload the tuning activities from operators!*

- Dynamic tuning will focus on:
  - Adjusting Linux kernel's buffers (for instance, controlling TCP window) via Sysctl
  - Setting TCP pacing for a steady data transfer based on network state:
    - *If there is only 24Gbps available in the end-to-end path, why try to increase the data rate above the available limit and face drops?*
  - Setting the TCP congestion control algorithm based on network state and hosts' capabilities

Tuning: Conceptual Examples

* TBD after evaluations

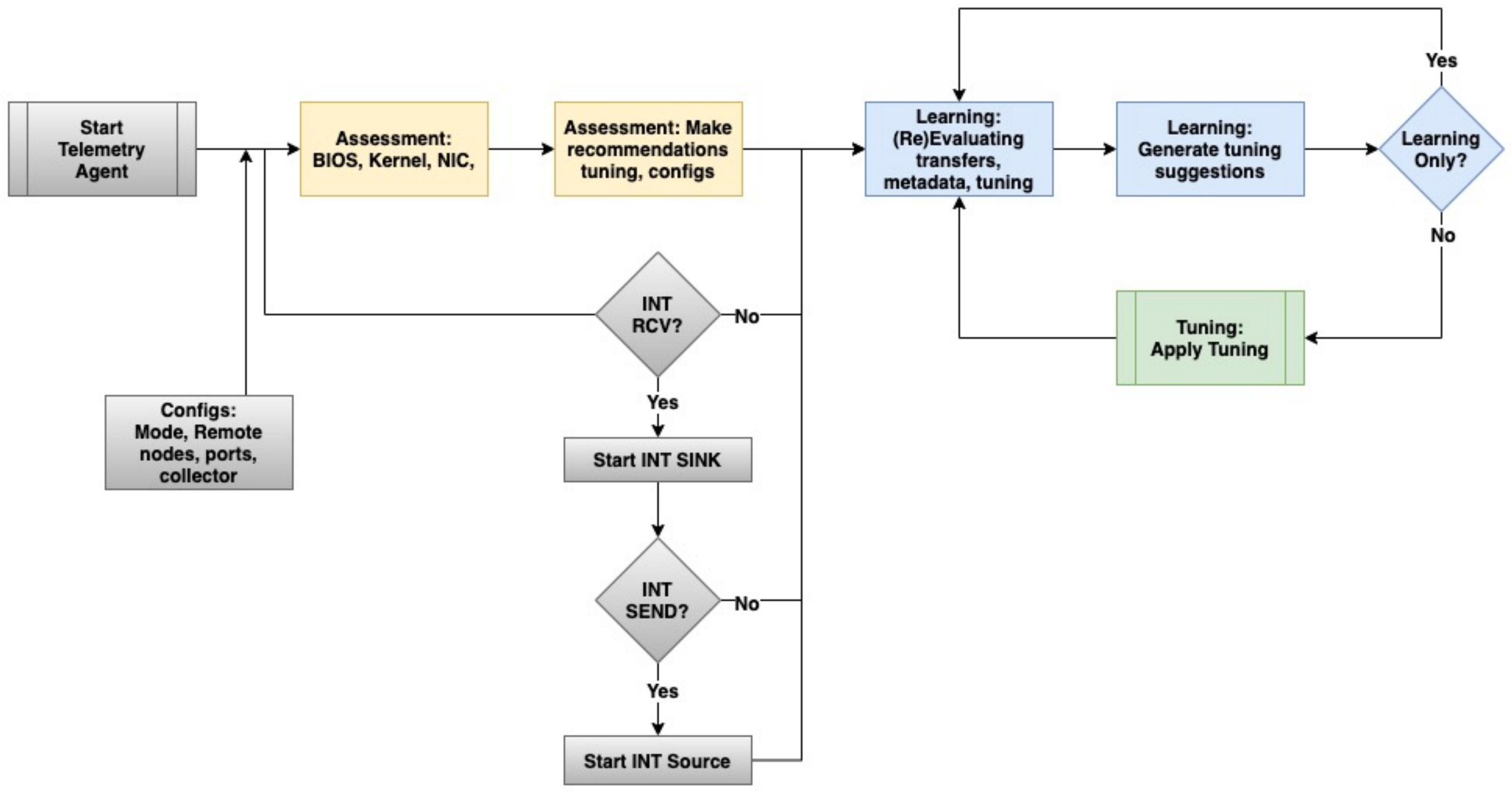
Apply:

- Default configuration is for *general* use (default installation)
- Under-provisioned tuning parameters (TCP congestion, write buffers)
- Data transfers' highest RTT > 50* ms
- Up: INT reports buffers < 2MB* and hop delay < 2* microseconds
- Down: INT reports buffers > 18MB* and hop delay > 50* microseconds

Don't Apply:

- Data transfers' highest RTT < 10* ms (DC, campus, metro network)
- Host's bandwidth already at max capacity
- Host's CPU operating at max capacity
- Host's Swap is used
- Host operating with BBR*

Building the Q-Factor Telemetry Agent - Workflow


Start Telemetry Agent
Assessment: BIOS, Kernel, NIC,
Assessment: Make recommendations tuning, configs
Learning: (Re)Evaluating transfers, metadata, tuning
Learning: Generate tuning suggestions
Learning Only?
Yes
No
Tuning: Apply Tuning
Configs: Mode, Remote nodes, ports, collector
INT RCV?
No
Yes
Start INT SINK
INT SEND?
No
Yes
Start INT Source

Q-Factor: Where are we now?

- Q-Factor is an NSF CC* Integration project planned for 2 years
  - Q-Factor is a little behind due to struggles with recruitment and the pandemic

- Milestones achieved:
  - Q-Factor testbed created with 4x 100G DTNs and 6x INT switches.
  - Adding INT header to outgoing packets using TC-eBPF.
  - Extracting INT telemetry using eBPF/XDP.
  - Q-Factor Telemetry Agent’s Tuning module is under active development
  - Next steps:
    - Extracting INT telemetry using Netronome programmable NICs
    - Evaluating tuning techniques following literature review
    - Developing observation-based learning techniques to consume queue occupancy and hop delay to mitigate bottlenecks

- Github: https://github.com/q-factor-project/

Science Drivers – Who benefits from Q-Factor?

- Distributed High Throughput Computing applications
    - LHCONE, OSG, EHT, …
- Real-Time, high-availability applications
    - Vera Rubin Observatory
- International Research Testbeds
    - Big Data Express, Pacific Research Platform, FABRIC, AutoGOLE/SENSE, …
- Network operators:
    - perfSONAR Community, ScienceDMZs, …

Engaging the community

- Q-Factor still has a long road ahead
- We are open for suggestions and collaborations
- Once Q-Factor is ready for production, it will be deployed at AmLight:
  - What use case would you like to see evaluated?
- Which direction should Q-Factor take next?
- Are you interested in participating in our discussions?
  - Join us at https://q-factor-project.slack.com

Team

Jeronimo Bezerra
(PI)
Florida International University

Dr. Julio Ibarra
(co-PI)
Florida International University

Dr. Richard Cziva
(co-PI)
Energy Sciences Network

Italo Valcy
Florida International University

Vasilka Chergarova
Florida International University

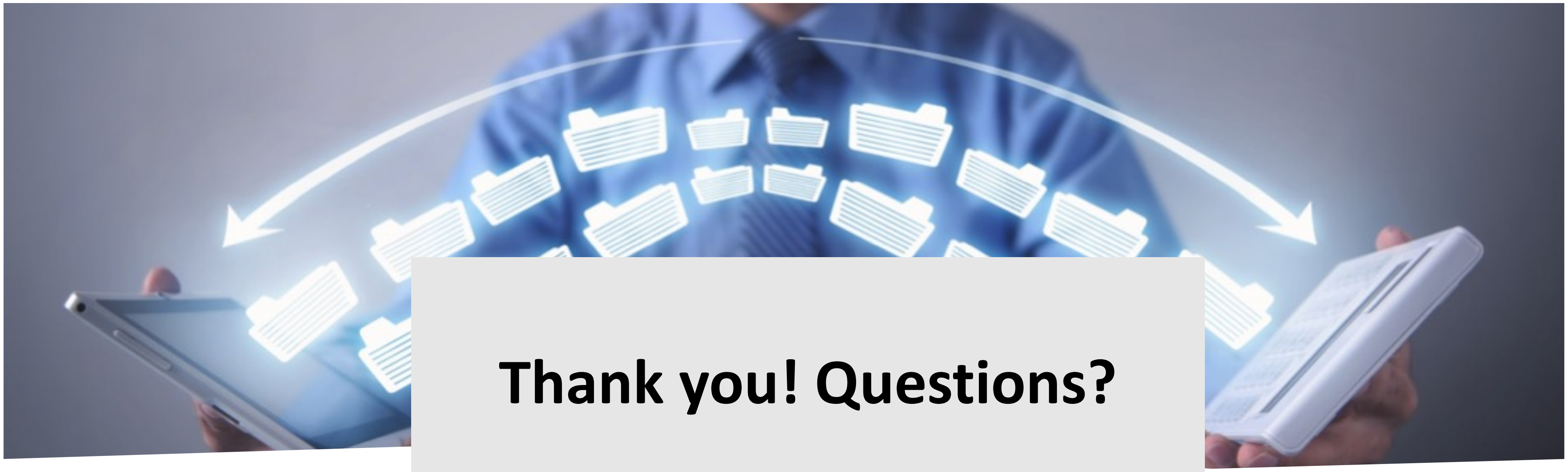
David Miranda
Florida International University

Eric Pouyoul
Energy Sciences Network

Sandesh Dhawaskar Sathyanarayana
Energy Sciences Network

David Ramirez
Florida International University

Dr. Ross Gegan
Energy Sciences Network



Thank you! Questions?

NSF CC*: Q-Factor
Real-time transfer optimization using programmable data planes

https://www.nsf.gov/awardsearch/showAward?AWD_ID=2018754

https://www.q-factor.io

Jeronimo Bezerra <jbezerra@fiu.edu>

Dr. Julio Ibarra <julio@fiu.edu>

Dr. Richard Cziva <Richard@es.net>