NSF CC*: Q-Factor

Real-time transfer optimization using programmable data planes

Jeronimo Bezerra, FIU, PI

Dr. Wenji Wu, ESnet, Co-PI

Dr. Julio Ibarra, FIU, Co-PI

NSF Program Director: Dr. Deep Medhi

Outline

- Motivation
- Goals
- Why now?
- Leveraging In-band Network Telemetry
- Deploying a Distributed Management Plane
- Building the Q-Factor Telemetry Agent
- Science Drivers
- Engaging the Community
- Team

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

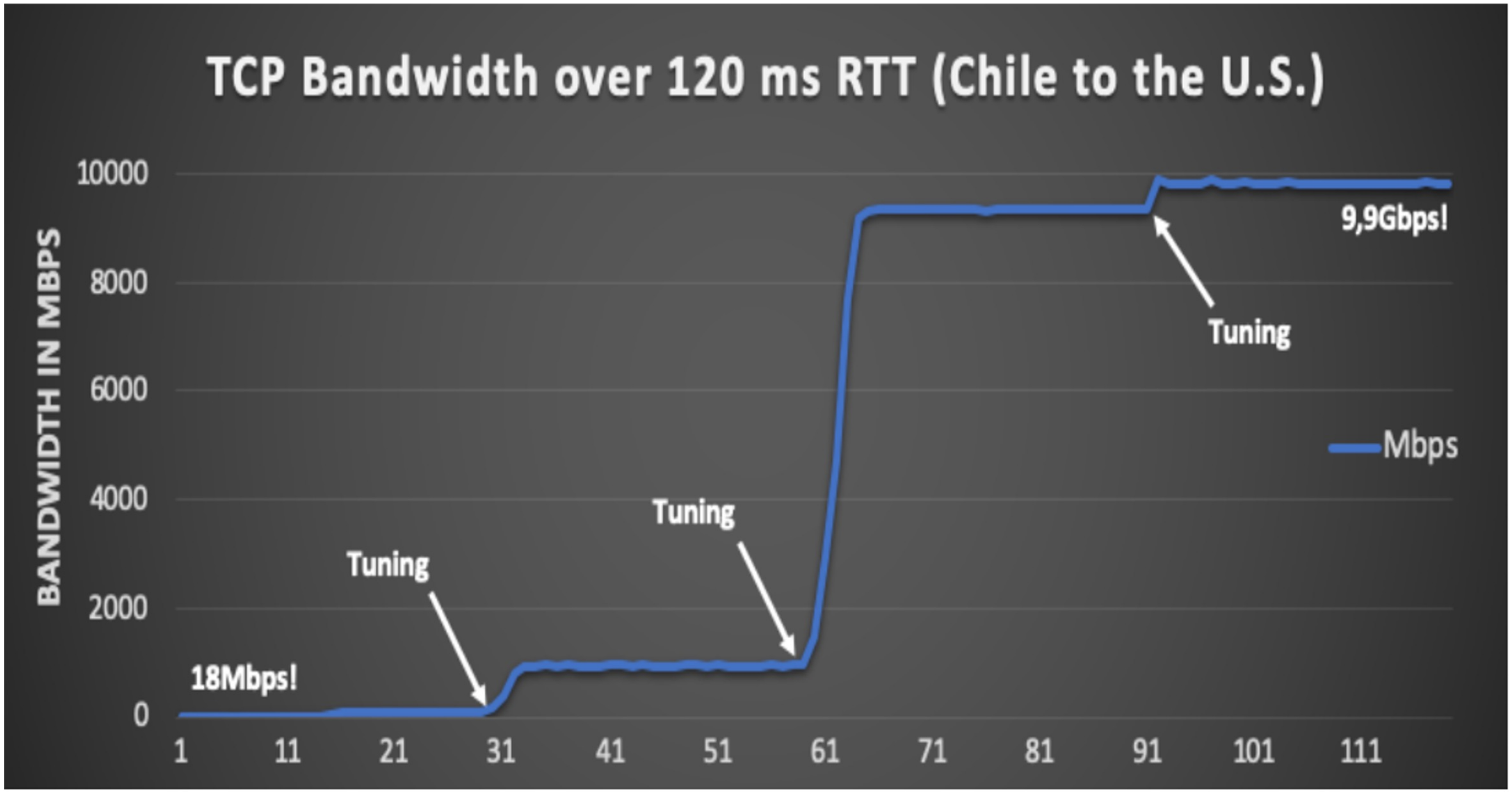
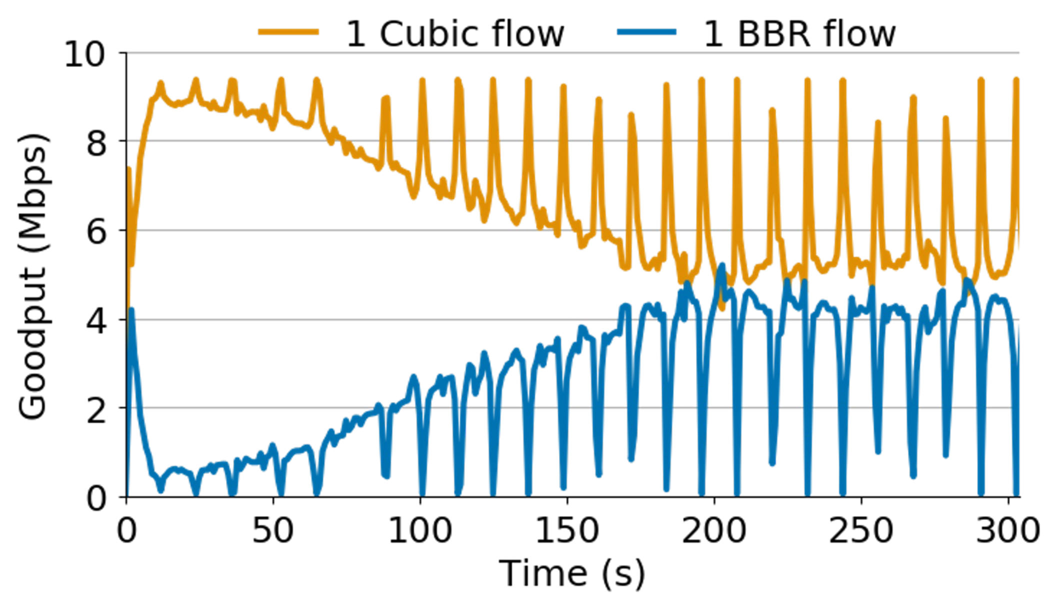


Figure 1 TCP Bandwidth over 120ms RTT when tuning is applied

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 only infers congestion



- *A packet is <u>dropped</u> 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)
  - …
  - Still, congestion control only infers network state
  - *What if the network forwarding devices could share the buffer utilization?*

Q-Factor's Goals

Offload tuning from IT operators	*Dynamically* tune hosts based on	*Leveraging* new telemetry technologies
• Telemetry agents on the hosts doing the tuning or making suggestions to operators	• Hardware capabilities • OS configuration • +RTT • +network conditions	• P4/In-band Network Telemetry • Programmable NICs • Linux eBPF & XDP

- **NSF Award #: 2018754**
- **Collaboration between Florida International University (FIU) and Energy Sciences Network (ESnet)**

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/buffer 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:

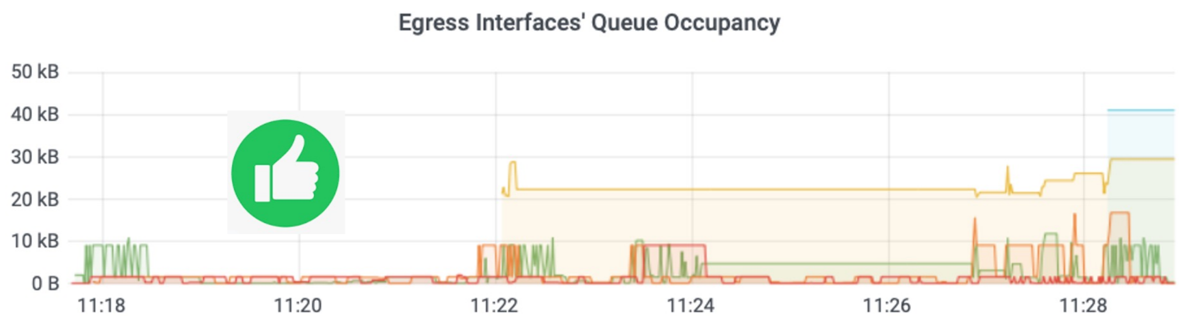
Translating: *you can track/monitor/evaluate EVERY single packet at line rate and real time.*

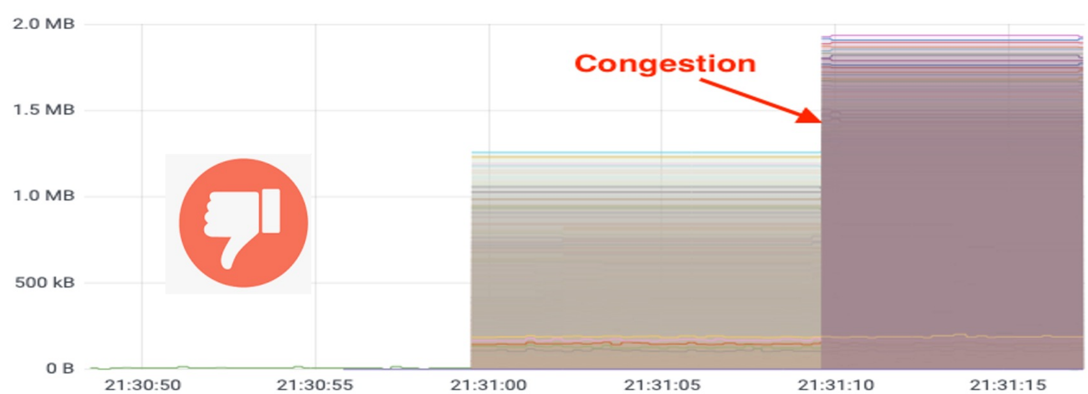
Examples of telemetry information added:

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

Leveraging INT – Mitigating Buffer/Queue Utilization

- 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).





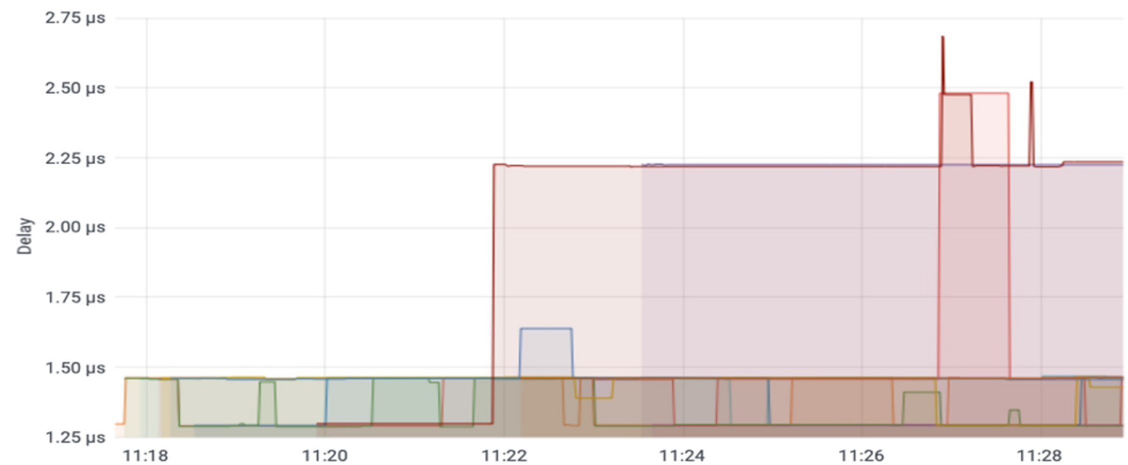
Leveraging INT – Observing Hop Delay & Jitter

Sources of jitter

Ideal Hop Delay:

- 1-10's Microsecond's delay

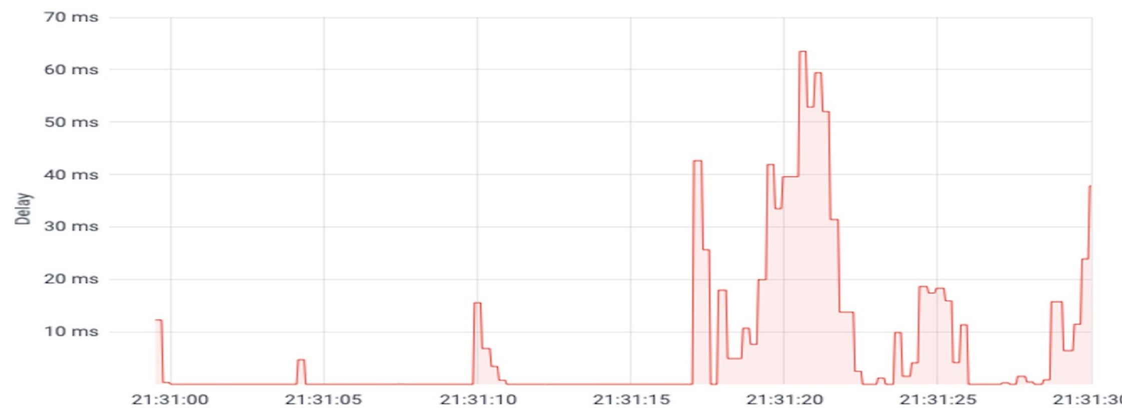
- Barely noticeable by users or applications in 10+ ms RTT environments



Hop Delay during a burst:

- 1000-10000's Microsecond's delay

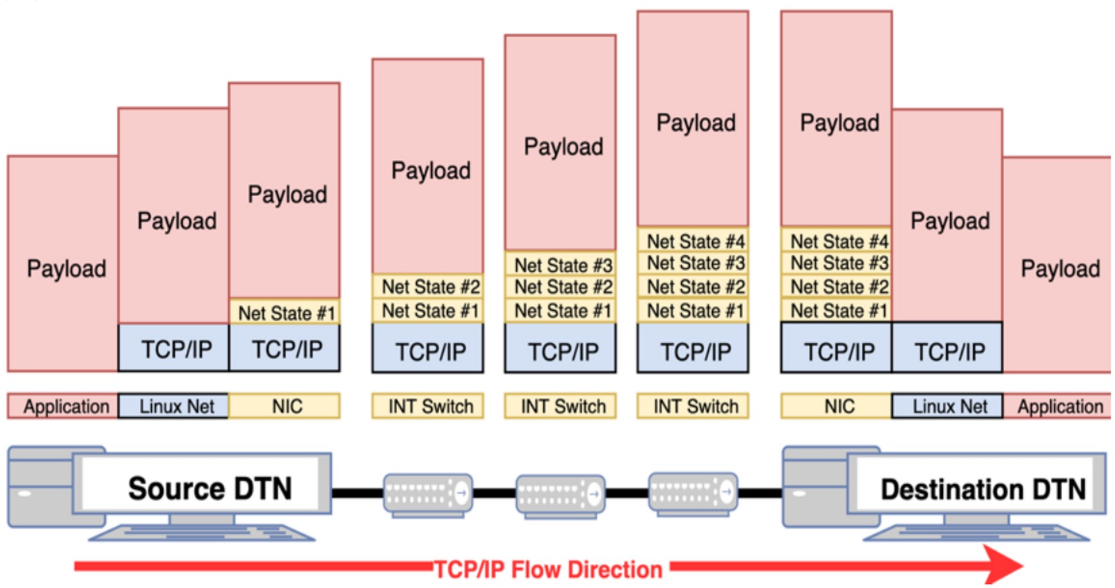
- Dramatically noticeable by users or applications for all environments



- P4/INT enables network operators to have even deeper visibility
  - 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 still have to infer the network state

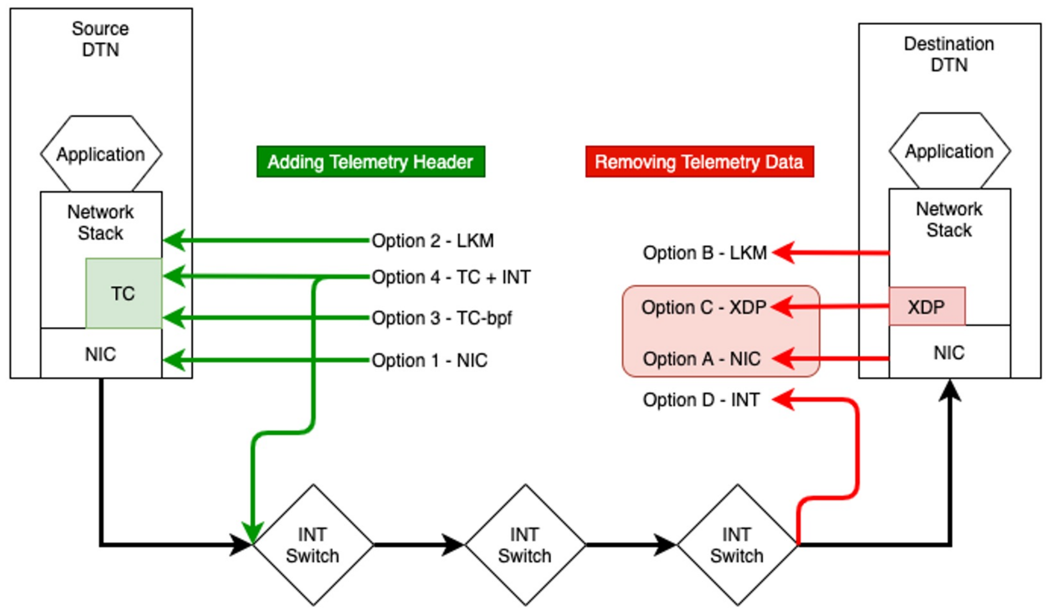
Q-Factor: Deploying a Distributed Management Plane

- 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?



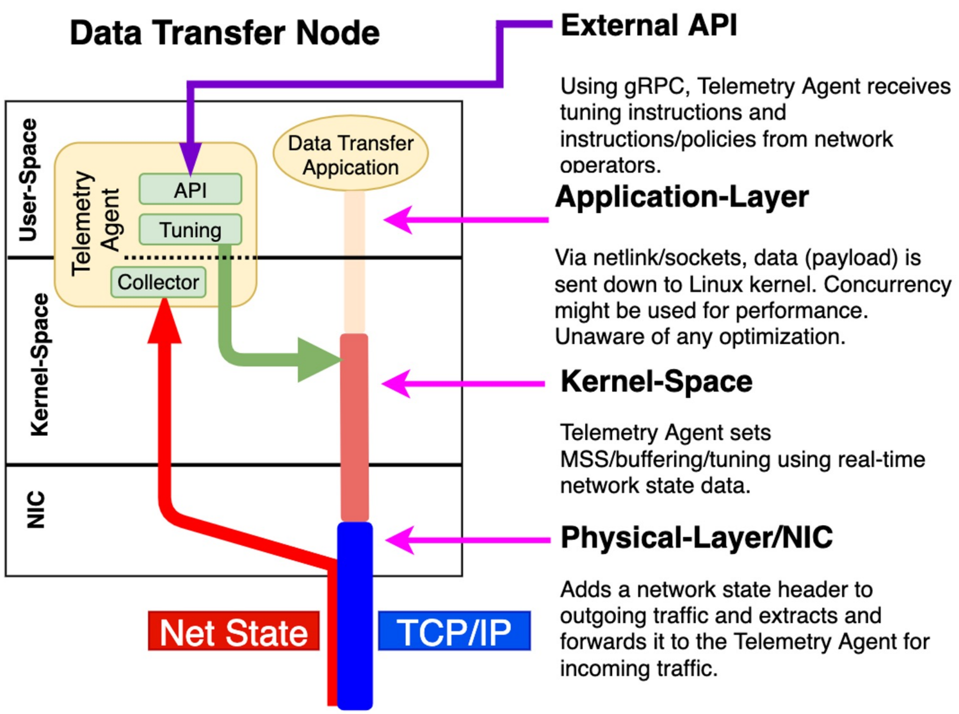
How will Q-Factor operate? [1/2]

- 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



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 Examples

* TBD after evaluations

Apply tuning when:

- Under-provisioned tuning parameters (TCP congestion, write buffers)
- Data transfers’ highest RTT > 50* ms
- Up: INT reports buffers < 1MB* and hop delay < 20* microseconds
- Down: INT reports buffers > 2MB* and hop delay > 100* microseconds

Don’t apply tuning when:

- 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

Q-Factor: Where are we now?

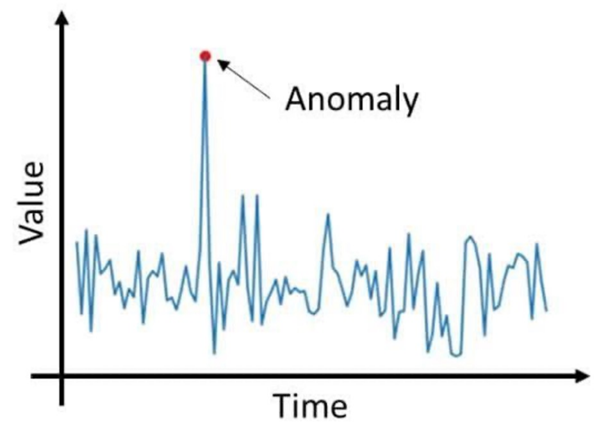
- Q-Factor is an NSF CC* Integration project planned for 2 years
  - Q-Factor is a little behind due to struggles with recruitment and the pandemic

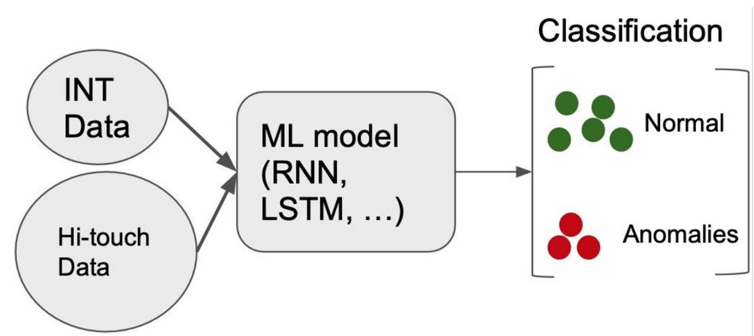
- Milestones achieved:
  - Extracting INT telemetry using eBPF/XDP and Netronome NICs at the destination at 40Gbps!
  - Q-Factor Telemetry Agent's Tuning module is ready for static host tuning.
  - Next steps:
    - Developing observation-based learning techniques to consume queue occupancy and hop delay to mitigate bottlenecks
    - Apply tuned accordingly to the observation-based learning techniques
    - Leveraging Machine Learning to detect complex anomalies.

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

Plus: ML-based Anomaly Detection using In Network Telemetry Data

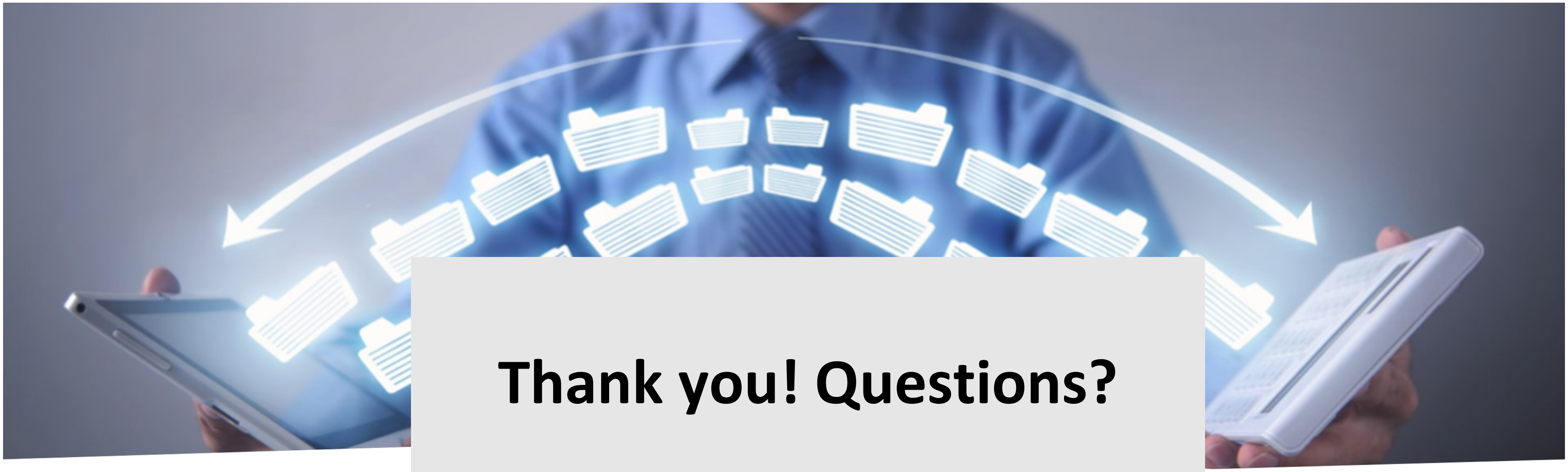
- Anomalies are events that **deviate** significantly from "normal" network traffic.
- Types of anomalies:
  - Security threats: DDoS attacks
  - Performance issues
    - Micro-congestion, link over-utilization, heavy flows, overloaded hosts, packet loss
- Current telemetry solutions (sFlow, NetFlow) send only a small sample of data for analysis for maintaining low overhead.
- Establishing a normal profile and detecting time critical events becomes hard or impossible.
- INT data provides detailed in-network information in real time.
  - Normal profile can be established using Machine Learning models trained on INT data enhanced with Esnet Hi-touch data.





Engaging the community

- Q-Factor still has a long road ahead (1.4 years)
- 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



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. Wenji Wu <wenji@es.net>

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