Enabling Silent Telemetry Data Transmission with InvisiFlow

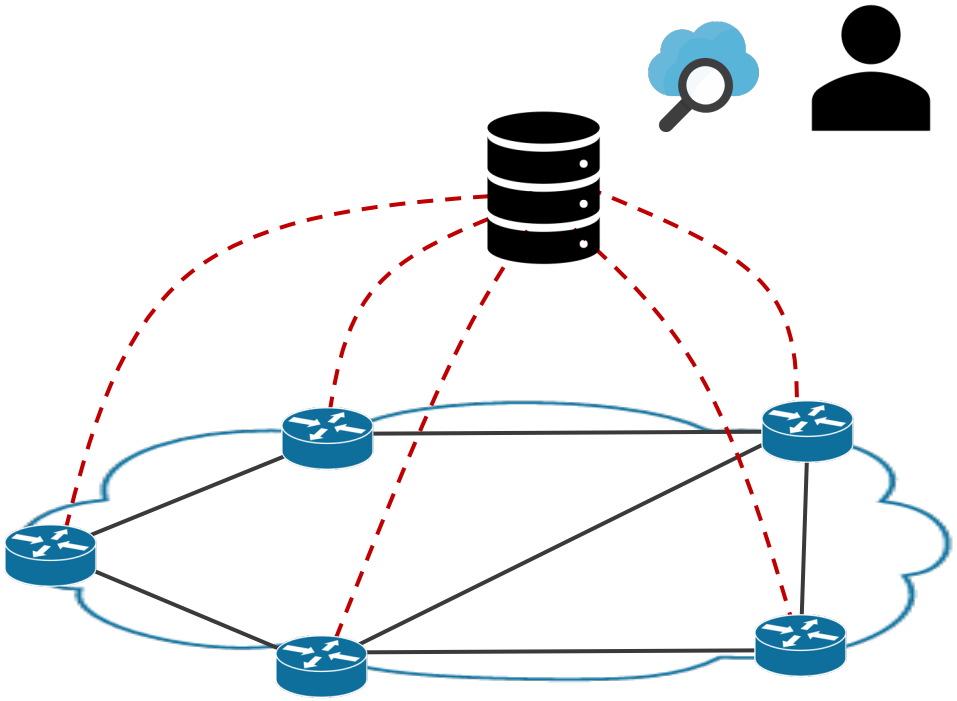
Yinda Zhang, Liangcheng Yu, Gianni Antichi, Ran Ben Basat, Vincent Liu

Microsoft Research

Outline for this talk

- Background and Motivation
- Existing Solutions and Limitations
- InvisiFlow Design
- Implementation and Evaluation
- Future work and Conclusion

Today's network are generating tons of telemetry data



Applications

Congestion Control: queue length in switches?
(e.g., HPCC [SIGCOMM'19])

Path Tracing: paths of flows?
(e.g., PathDump [OSDI'16])

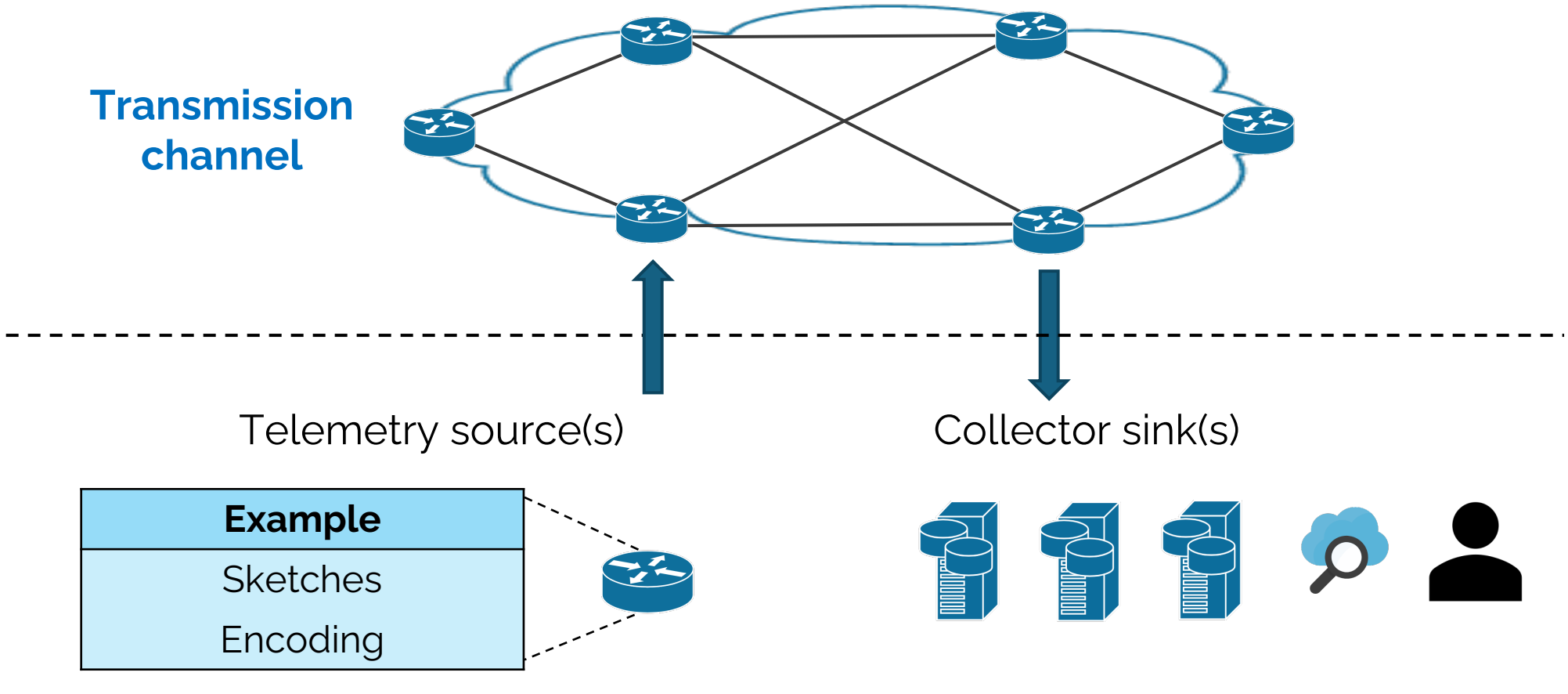
Load balancing: utilization of links?
(e.g., CONGA [SIGCOMM'14])

Today's network are generating tons of telemetry data

- Telemetry data in **smaller timescales**
  - Bursts in less than 10s of microseconds *[IMC'17]*
  - Need of per-packet in-band metrics *[SIGCOMM'19]*
- **More kinds** of telemetry data
  - Switch-related *(e.g., buffer statistics)*
  - Flow-related *(e.g., flow size)*

Workflow of telemetry systems

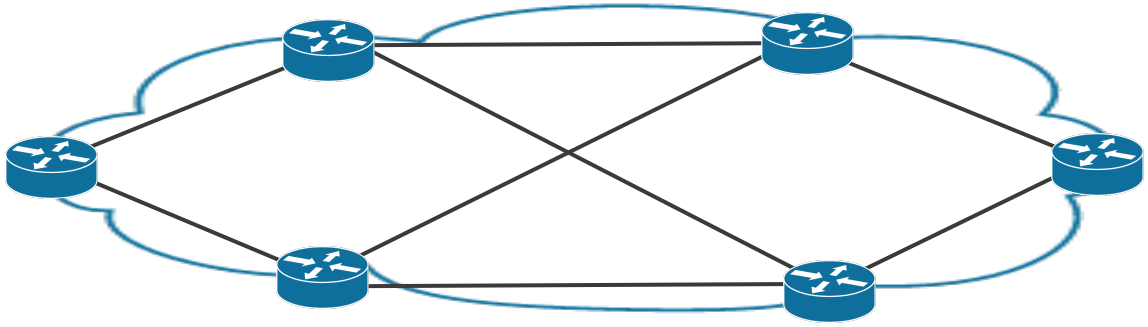
Often, telemetry and user traffic **multiplex** the same network



Telemetry systems are facing a tradeoff

Often, telemetry and user traffic **multiplex** the same network

Transmission channel



- ❑ ***Maximize the sustainable throughput*** for telemetry data
- ❑ ***Near-zero overhead*** on user traffic

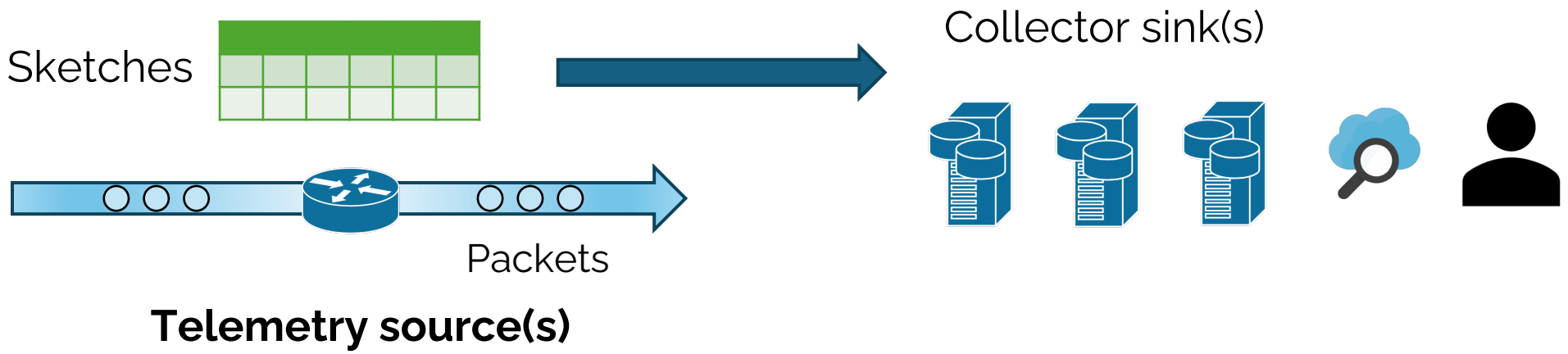
Outline for this talk

- Background and Motivation
- Existing Solutions and Limitations
- InvisiFlow Design
- Implementation and Evaluation
- Future work and Conclusion

Reduce size of data in telemetry sources

（UnivMon [SIGCOMM'16], NitroSketch [SIGCOMM'19], CocoSketch [SIGCOMM'21]）

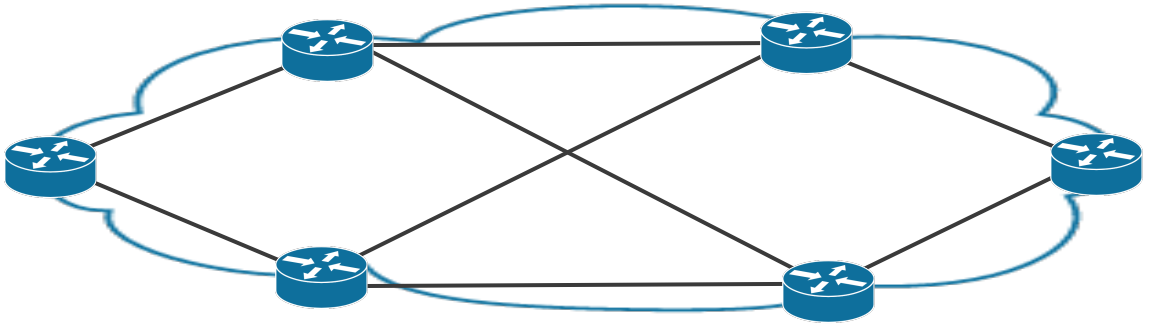
- Sacrifice **generality & accuracy** for the required bandwidth
  - **Generality:** Different kinds of sketches for different tasks
  - **Accuracy:** *PINT* could miss > 60% of flows path to limit bandwidth overhead to < 2%



Transmission channel still needs improvement

- ***Maximize the sustainable throughput*** for telemetry data
- ***Near-zero overhead*** on user traffic

Transmission channel

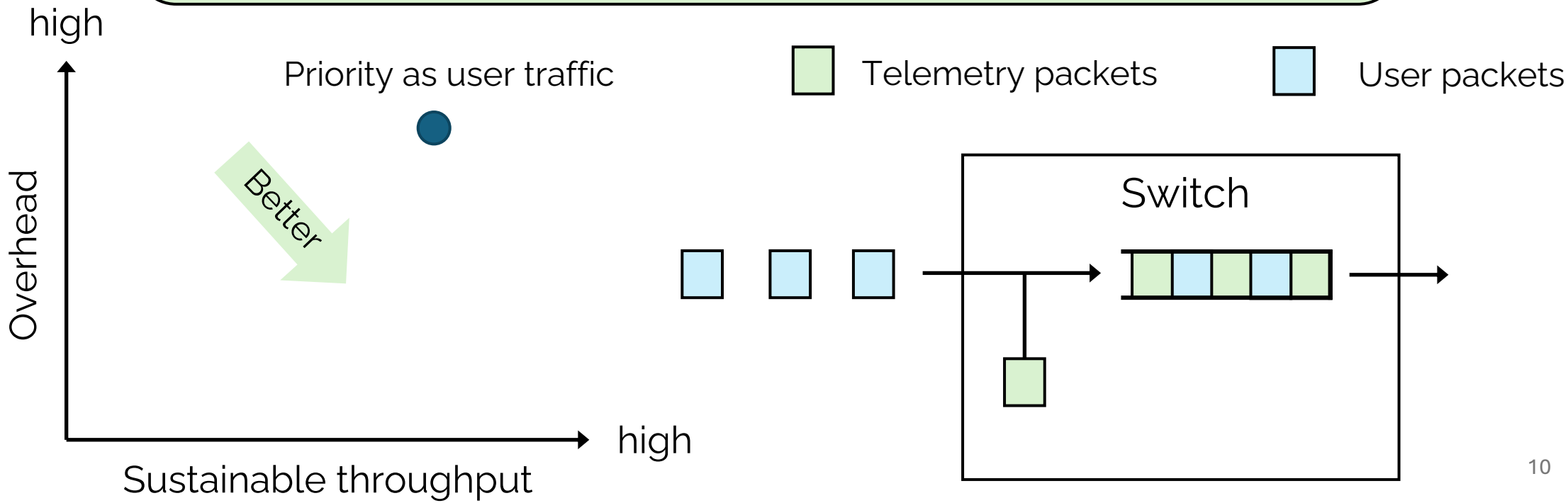


Often, telemetry and user traffic **multiplex** the same network

High sustainable throughput but high overhead

- ***Maximize the sustainable throughput*** for telemetry data
- ***Near-zero overhead*** on user traffic

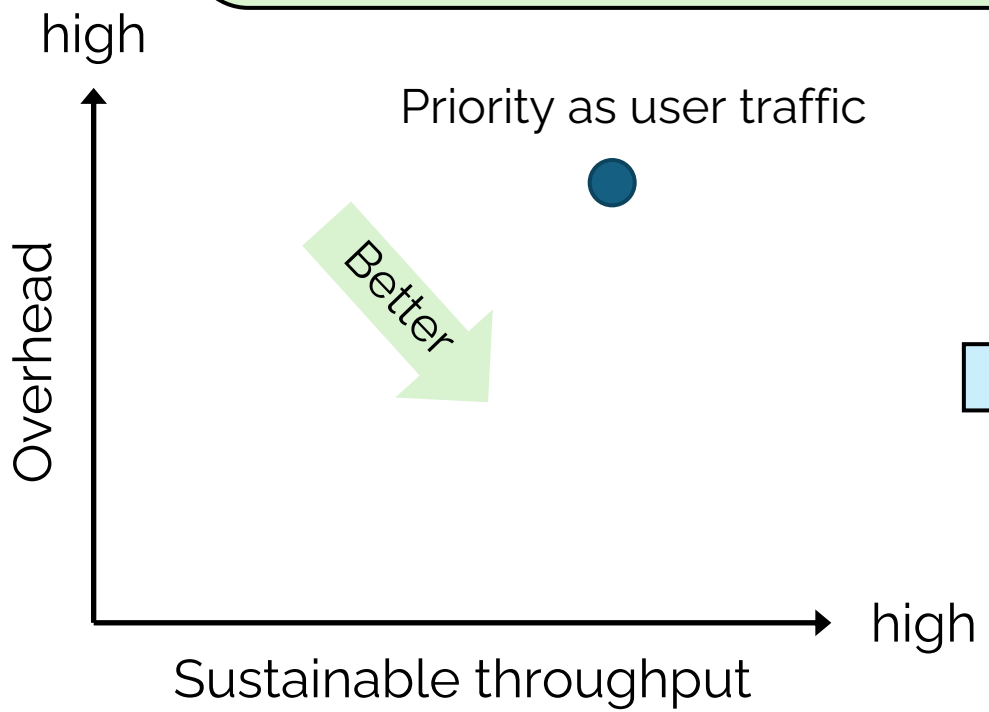
~19% increase in user traffic flow completion time

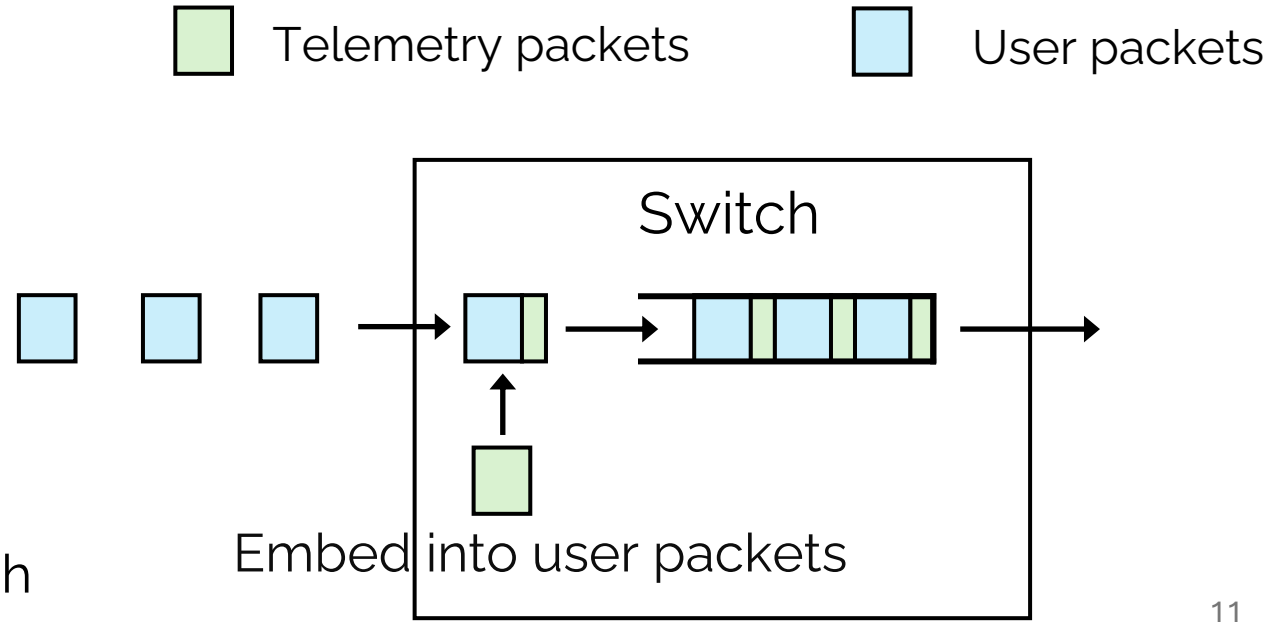


High sustainable throughput but high overhead

- ***Maximize the sustainable throughput*** for telemetry data
- ***Near-zero overhead*** on user traffic

~20% degradation of application-level throughput



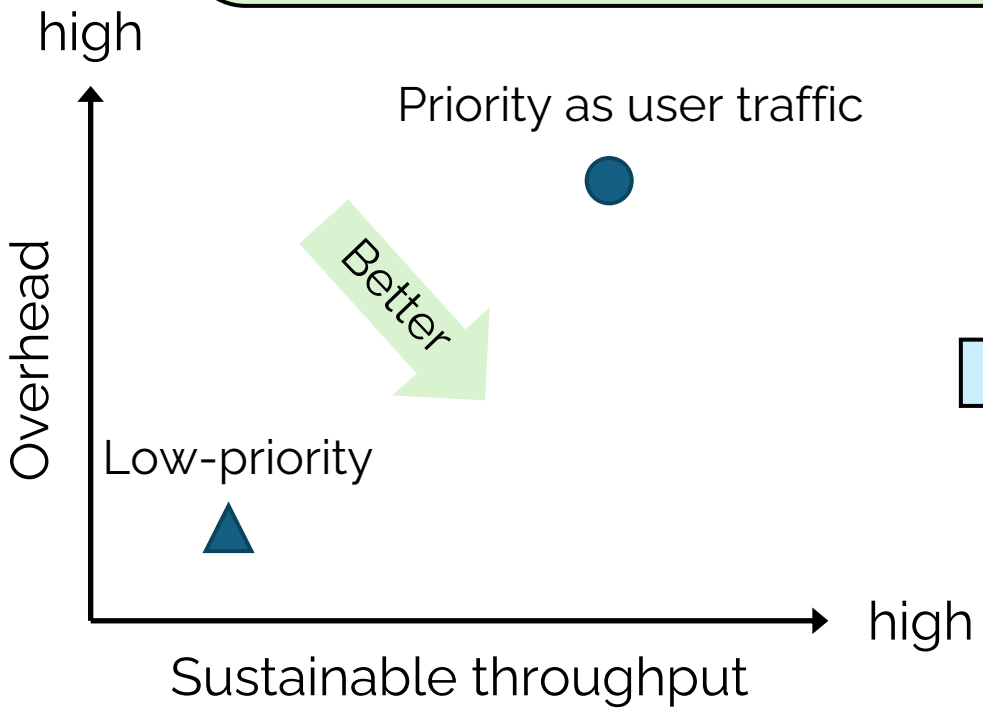


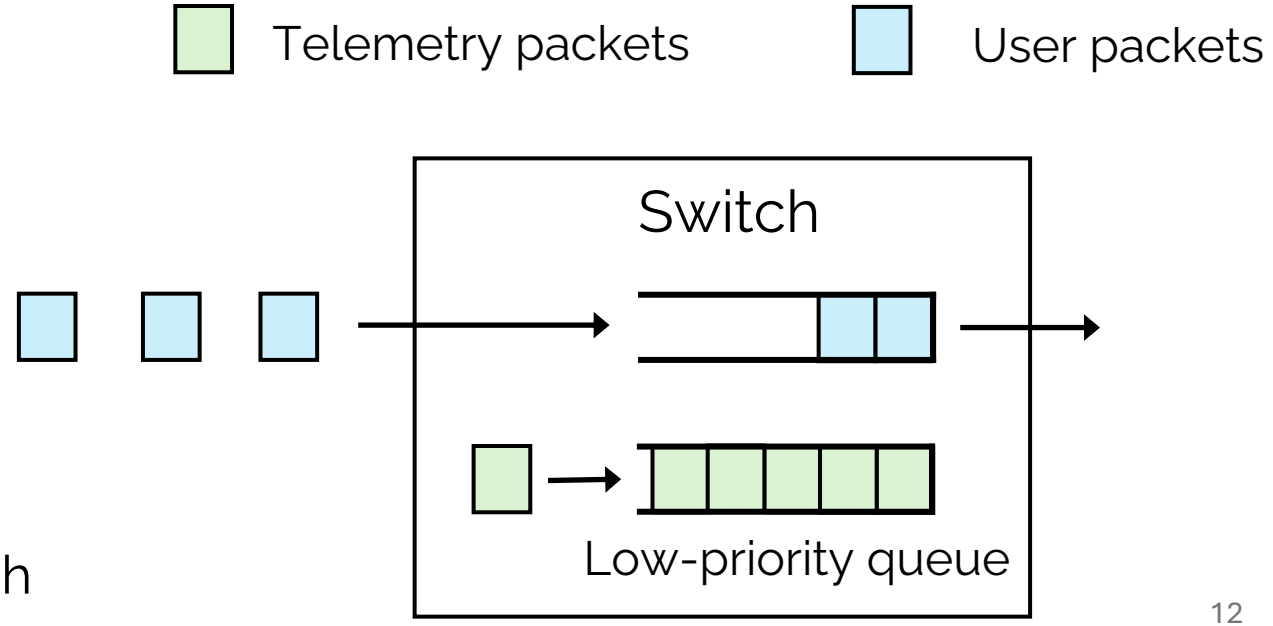
Low overhead but low sustainable throughput

(Planck [SIGCOMM'14], Everflow [SIGCOMM'15])

- ***Maximize the sustainable throughput*** for telemetry data
- ***Near-zero overhead*** on user traffic

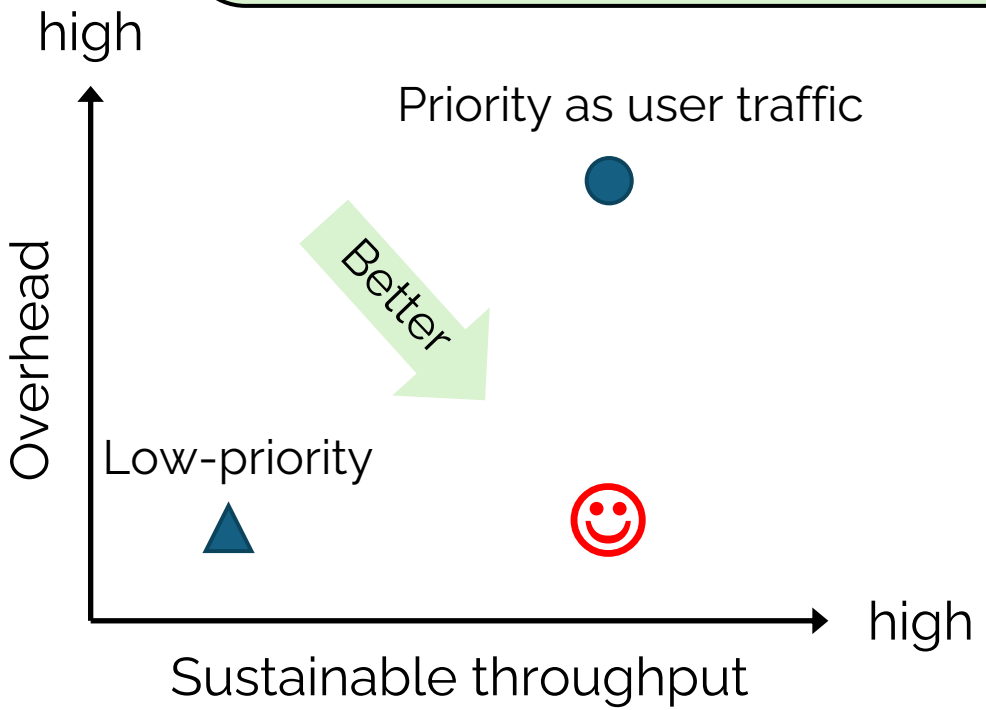
11% missing paths for path tracing





Transmission channel are facing a tradeoff

- ***Maximize the sustainable throughput*** for telemetry data
- ***Near-zero overhead*** on user traffic



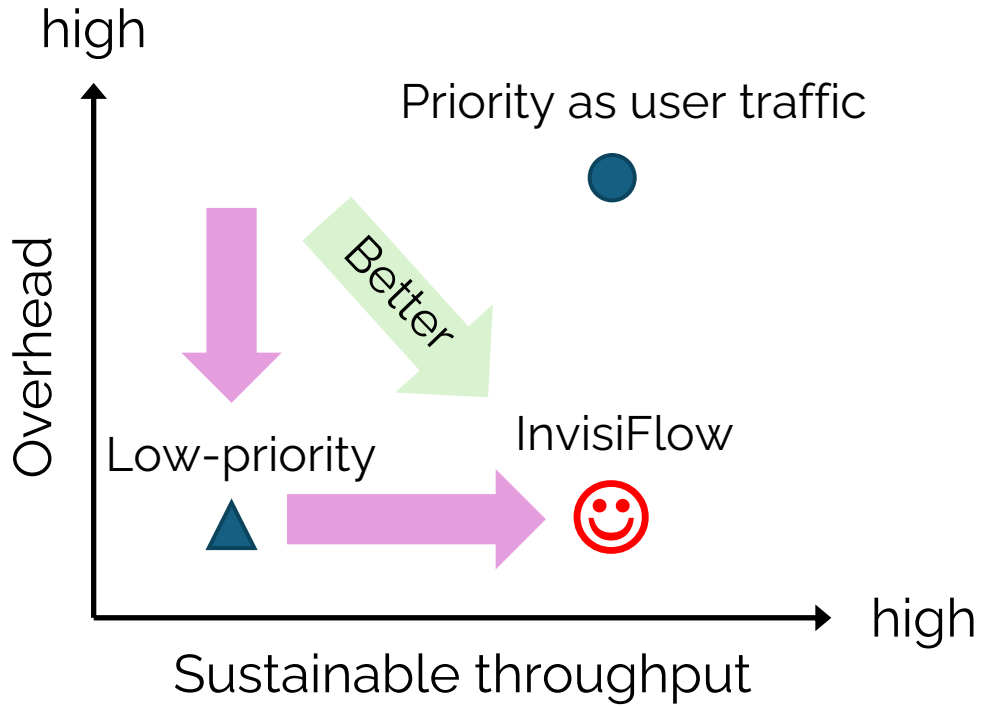
Is it possible to meet both requirements?

Outline for this talk

- Background and Motivation
- Existing Solutions and Limitations
- InvisiFlow Design
  - Intuition behind
  - Architecture and Details
- Implementation and Evaluation
- Future work and Conclusion

Two design principles of InvisiFlow

- Prioritize user traffic
  - Achieve near-zero overhead on user traffic
- Seek out spare capacity over the network
  - Reduce loss rate of telemetry data



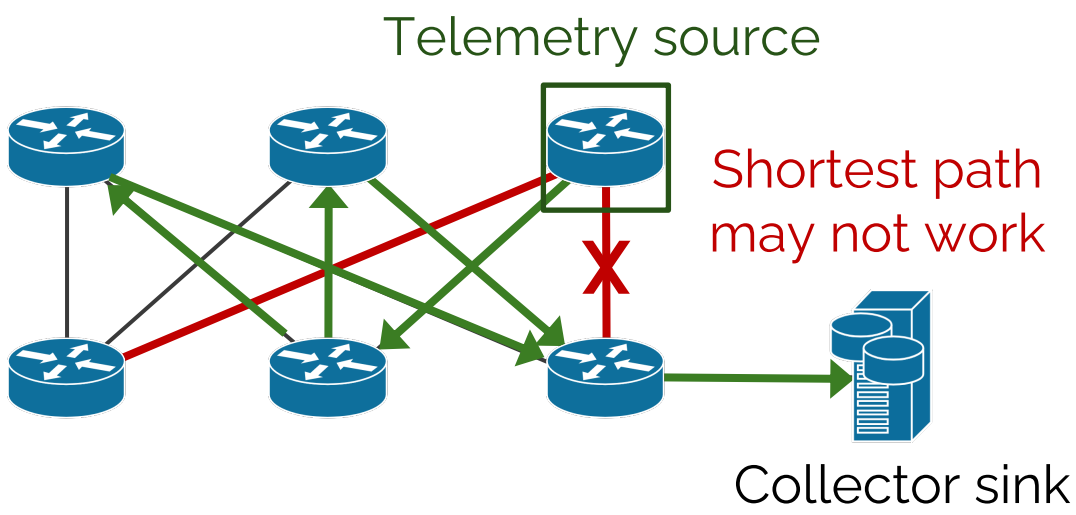
Where is the spare capacity?

Spare capacity often exists over the network

Where is the spare capacity?

Links fully occupied by user traffic

Available links for telemetry data



There are often available paths due to the ***redundancy*** of the data center topology and ***bursty*** user traffic

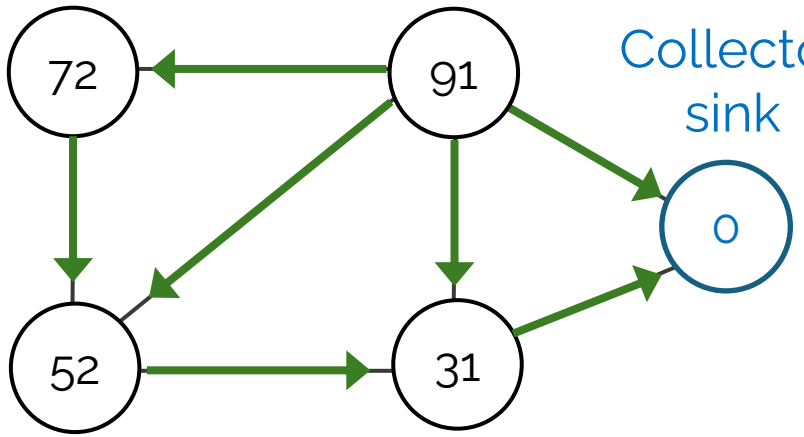
How can we find all available paths given the dynamic arrival pattern of user traffic?

Congestion gradient-based transmission

congestion gradient

Forward telemetry data based on the ~~shortest path~~

k Switch with k% telemetry buffer usage



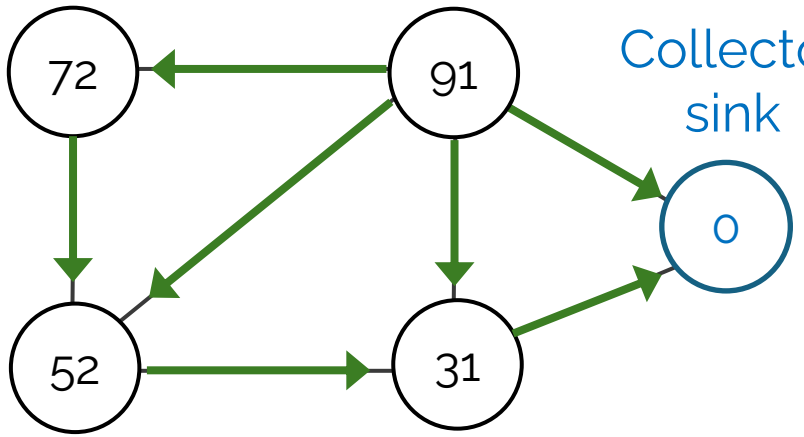
Congestion gradient: Difference in telemetry buffer usage between neighboring switches

Congestion gradient-based transmission: Every switch sends data to ***all*** neighbors with ***lower*** *telemetry buffer usage*

Congestion gradient-based transmission

- Prior theories prove that *congestion gradient-based transmission* can
  - **Maximize the sustainable throughput**
  - **Robust** to diverse topology and dynamic traffic pattern

k Switch with k% telemetry buffer usage



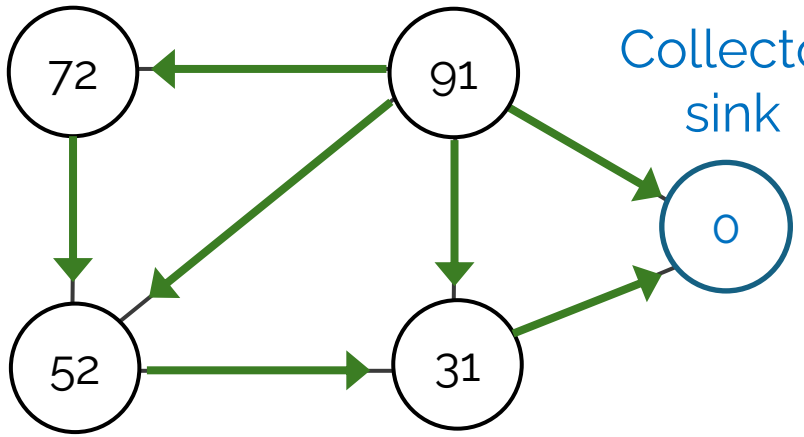
Congestion gradient: Difference in telemetry buffer usage between neighboring switches

Congestion gradient-based transmission: Every switch sends data to ***all*** neighbors with ***lower*** *telemetry buffer usage*

Implementing InvisiFlow is challenging

- Calculate gradients over time to capture dynamic patterns
- Limitations in packet processing pipeline (e.g., early binding of egress ports)

k Switch with k% telemetry buffer usage



Congestion gradient: Difference in telemetry buffer usage between neighboring switches

Congestion gradient-based transmission: Every switch sends data to ***all*** neighbors with ***lower*** *telemetry buffer usage*

Outline for this talk

- Background and Motivation
- Existing Solutions and Limitations
- InvisiFlow Design
  - Intuition behind
  - Architecture and Details
- Implementation and Evaluation
- Future work and Conclusion

Pull-based transmission channel in InvisiFlow

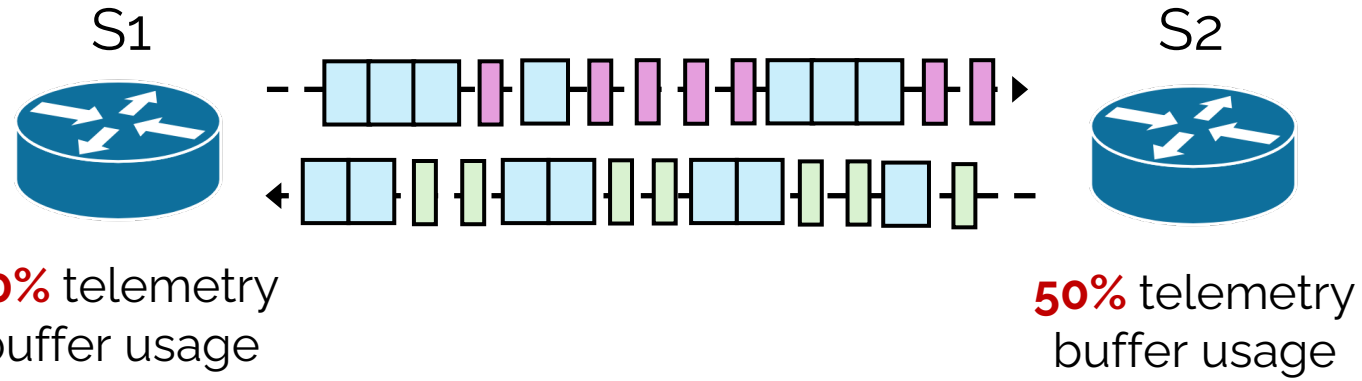
User packets

Pull requests

Carry the buffer usage

Telemetry packets

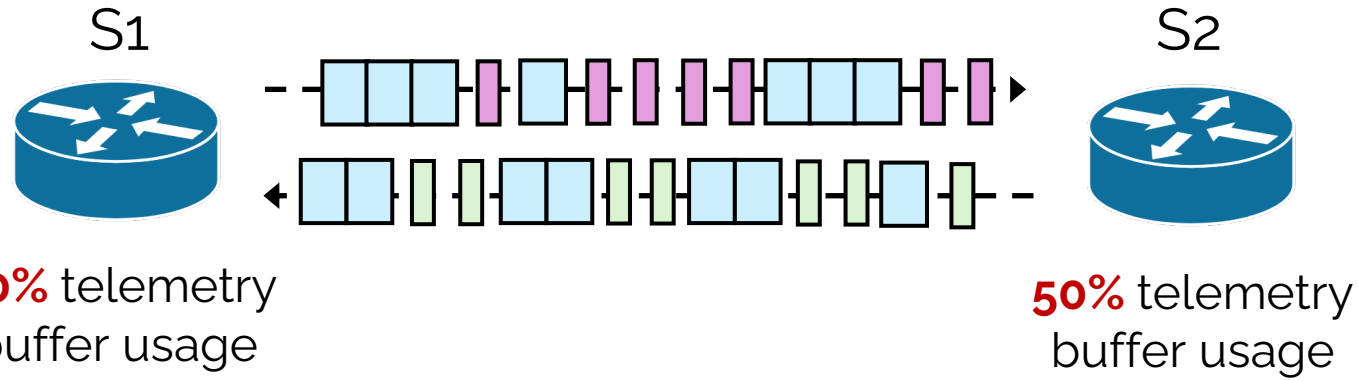
Packet generator in the programmable switches allow us to inject ***low-priority pull requests*** with **near-zero overhead**



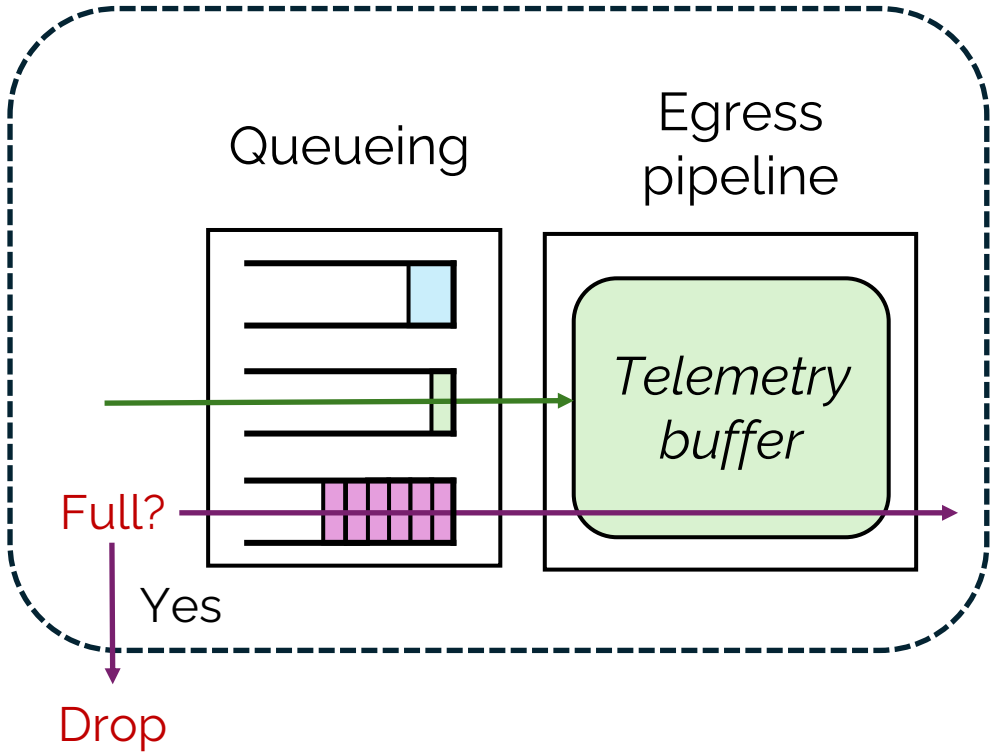
Pull-based transmission channel in InvisiFlow

User packets Pull requests Telemetry packets

User **free ports** for telemetry packets
Free ports: ports not typically used by user traffic
(e.g., recirculation ports)

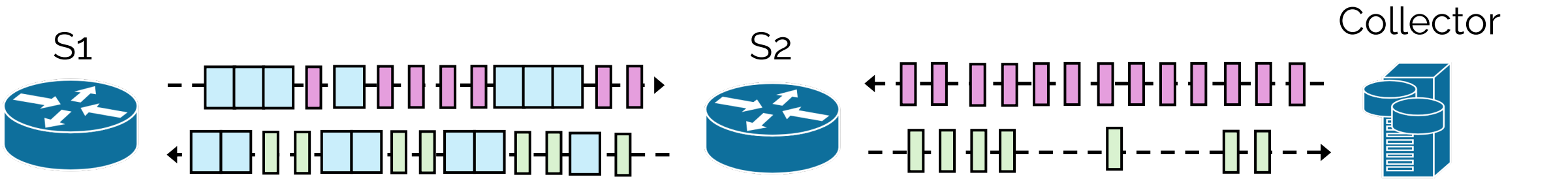


Send telemetry packets if the other sides' buffer usage is **lower**



Pull-based transmission channel in InvisiFlow

User packets Pull requests Telemetry packets



30% telemetry buffer usage

50% telemetry buffer usage

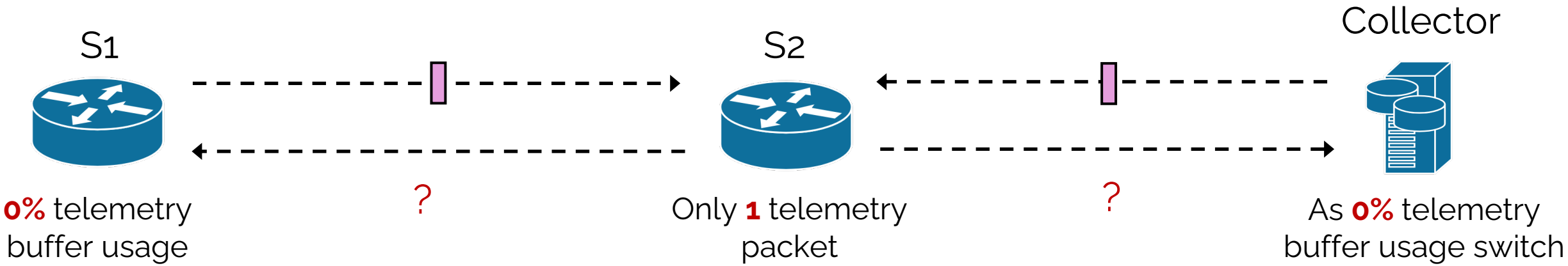
As **0%** telemetry buffer usage switch

Send telemetry packets if the other sides' buffer usage is **lower**

Zero congestion gradient

User packets Pull requests Telemetry packets

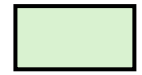
Oscillate between S1 and S2
given **zero congestion gradient**



Adding a ‘slope’ to prevent zero congestion gradient


User packets


Pull requests


Telemetry packets

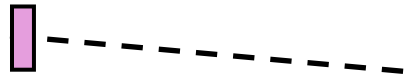
Oscillate between S1 and S2
given **zero congestion gradient**

S1

0% telemetry
buffer usage

S2

Only **1** telemetry
packet



Collector

As **0%** telemetry
buffer usage switch

Successfully pull data if and only if
usage + ***distance*** is smaller

Outline for this talk

- Background and Motivation
- Existing Solutions and Limitations
- InvisiFlow Design
- Evaluation
- Future work and Conclusion

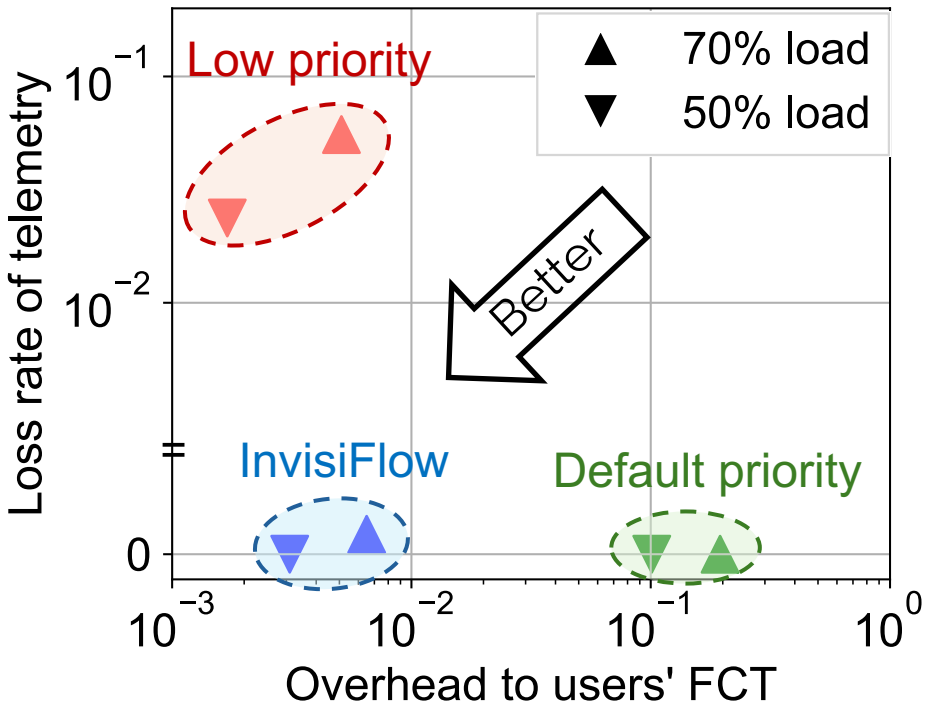
Evaluation setup

- **Testbed**
  - Leaf-spine topology with 10Gbps links
- **NS-3 Simulation**
  - Fat-tree topology with 100Gbps links
  - Hadoop + ML workload

Simulation setting

- **Transmission Channel Baseline**
  - Default-priority UDP (priority same as user traffic)
  - Low-priority UDP
- **Different Telemetry Applications**
  - Path tracing *(NetSeer [SIGCOMM'20])*
  - Load imbalance detection *(NetFlow)*
  - Per-flow accounting *(ApproSync [ICNP'20])*
  - Packet drop notification (*NetSeer [SIGCOMM'20])*

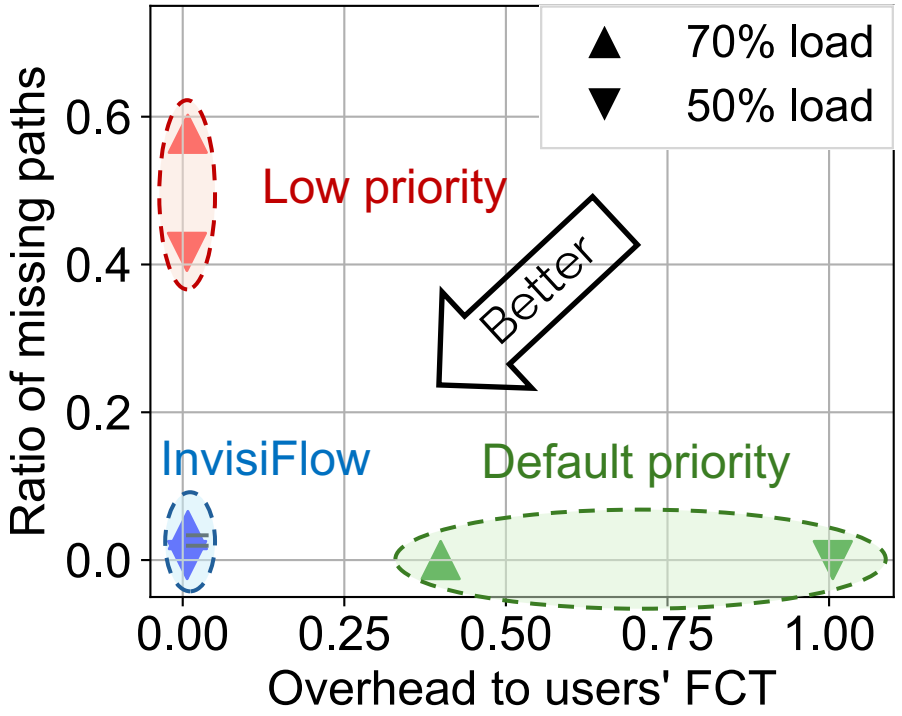
InvisiFlow meets both requirements



- *InvisiFlow* achieves both **near-zero telemetry loss** and **near-zero overhead on user traffic**
- *Low-priority UDP* suffers from **telemetry loss**, while *default-priority UDP* **increase user FCT**

InvisiFlow is robust to different settings

- Load imbalance (*faulty configuration*)
- Asymmetric topology *(failed links)*



Tradeoff for path tracing applications under asymmetry topology

Conclusions

- *Sustainable throughput for telemetry* vs. *Overhead on user*
- InvisiFlow for transmission channel
  - *Congestion gradient* to explore unused network bandwidth
  - Low-priority pull-based communication substrate
- Provide *near-zero loss of telemetry* and *little-to-no impact on user*
  - Robust to different settings and workloads

Source code: https://github.com/eniac/InvisiFlow