

TCP's Third Eye: Leveraging eBPF for Telemetry-Powered Congestion Control

Jörn-Thorben Hinz Vamsi Addanki Csaba Györgyi
Theo Jepsen (Intel) Stefan Schmid



Workshop on eBPF and Kernel Extensions
SIGCOMM 2023

A Host's Requirements

- Reliable transmission
- In-order data delivery
- Interoperability



A Host's Requirements

- Reliable transmission
- In-order data delivery
- Interoperability

The host requires TCP!



A Host's Goals

- High throughput, low delay
- Quick flow completion
- No packet drops



A Host's Goals

- High throughput, low delay
- Quick flow completion
- No packet drops

They require the network to not be congested!



Avoiding Network Congestion

TCP congestion control (CC):

- Attempts to avoid network overload
- Limits number of unacknowledged packets (congestion window, CWND)
- Part of the host network stack



Avoiding Network Congestion

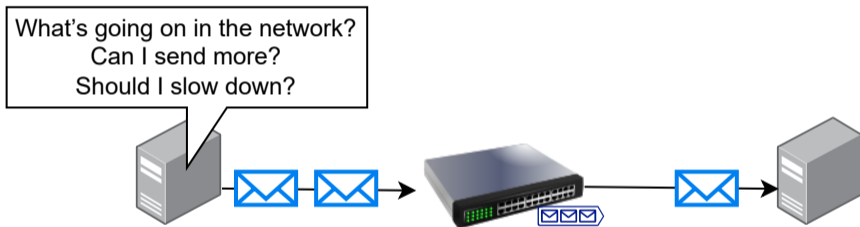
TCP congestion control (CC):

- Attempts to avoid network overload
- Limits number of unacknowledged packets (congestion window, CWND)
- Part of the host network stack

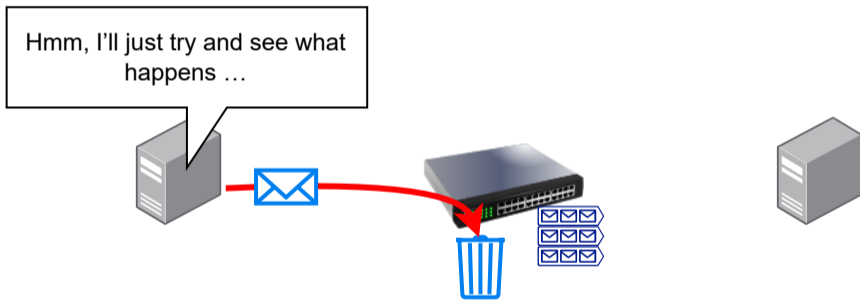
But how does a host know about the network load?



Lack of Sight

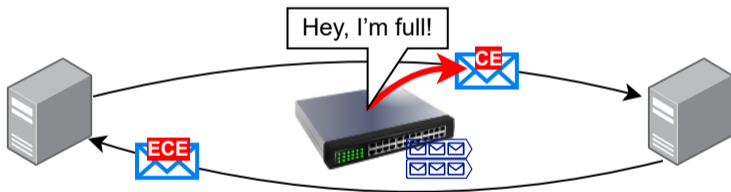


Lack of Sight: Delayed Feedback



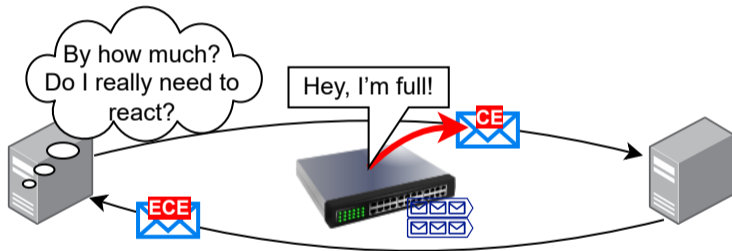
Lack of Sight: Imprecise Feedback

- Only standardized signal: ECN (Explicit Congestion Notification)



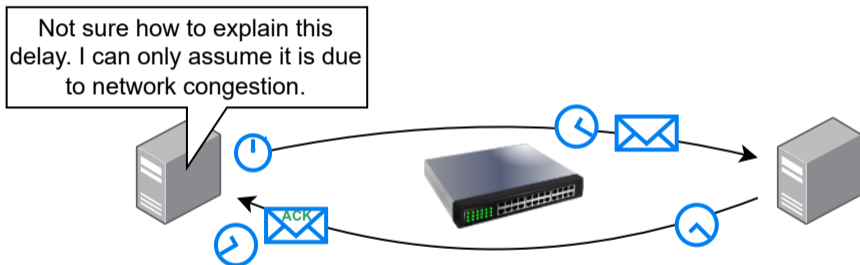
Lack of Sight: Imprecise Feedback

- Only standardized signal: ECN (Explicit Congestion Notification)



- Just binary information

Lack of Sight: Misleading Feedback

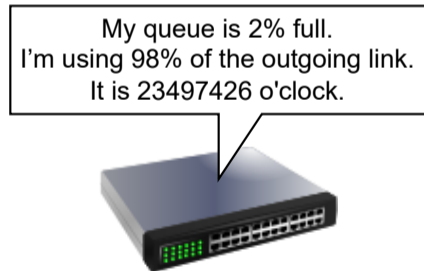


Enhancing Hosts' Sight: Telemetry

- Enhanced network feedback: In-band network telemetry (INT)

Enhancing Hosts' Sight: Telemetry

- Enhanced network feedback: In-band network telemetry (INT)
- Switch metrics useful for CC:
 - Queue depth
 - Link utilization
 - Timing information



Enhancing Hosts' Sight: Telemetry

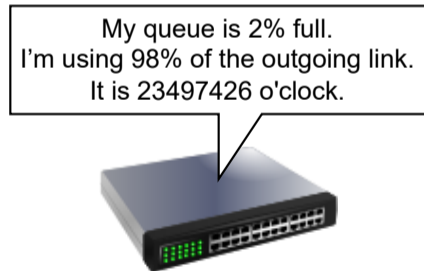
- Enhanced network feedback: In-band network telemetry (INT)
- Switch metrics useful for CC:
 - Queue depth
 - Link utilization
 - Timing information
- Telemetry ...

My queue is 2% full.
I'm using 98% of the outgoing link.
It is 23497426 o'clock.



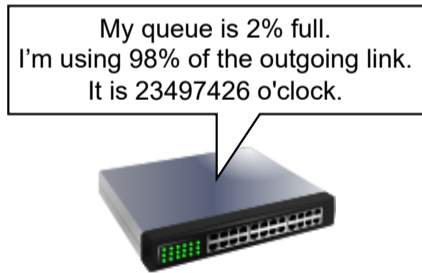
Enhancing Hosts' Sight: Telemetry

- Enhanced network feedback: In-band network telemetry (INT)
- Switch metrics useful for CC:
 - Queue depth
 - Link utilization
 - Timing information
- Telemetry ...
 - ~~Misleading~~
Is directly related to network load



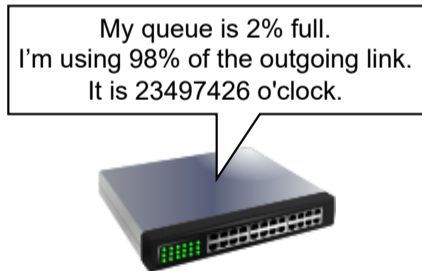
Enhancing Hosts' Sight: Telemetry

- Enhanced network feedback: In-band network telemetry (INT)
- Switch metrics useful for CC:
 - Queue depth
 - Link utilization
 - Timing information
- Telemetry ...
 - ~~Misleading~~
Is directly related to network load
 - ~~Imprecise~~
Provides rich, multi-bit metrics

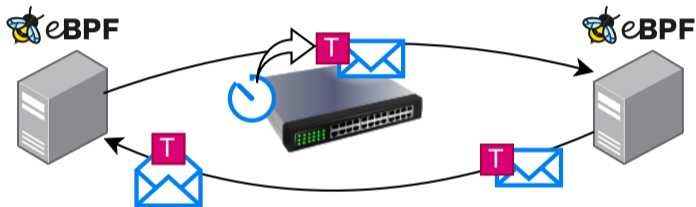


Enhancing Hosts' Sight: Telemetry

- Enhanced network feedback: In-band network telemetry (INT)
- Switch metrics useful for CC:
 - Queue depth
 - Link utilization
 - Timing information
- Telemetry ...
 - ~~Misleading~~
Is directly related to network load
 - ~~Imprecise~~
Provides rich, multi-bit metrics
 - ~~Delayed~~
Is frequently updated



The Third Eye: Congestion Control + INT



We want to implement this in a practical, deployable way!

Challenge: Delivering INT to Hosts

- No common or standardized INT solution available

Challenge: Delivering INT to Hosts

- No common or standardized INT solution available, **yet**
- Various standardization efforts, e.g., recent RFC drafts “Congestion Signaling (CSIG)”, “Advanced Explicit Congestion Notification”

Challenge: Delivering INT to Hosts

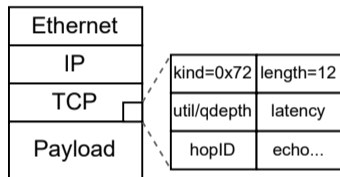
- No common or standardized INT solution available, yet
- Various standardization efforts, e.g., recent RFC drafts “Congestion Signaling (CSIG)”, “Advanced Explicit Congestion Notification”
- Other approaches around INT use . . .
 - Custom protocols
 - Non-standard protocol modifications

Challenge: Delivering INT to Hosts

- No common or standardized INT solution available, yet
- Various standardization efforts, e.g., recent RFC drafts “Congestion Signaling (CSIG)”, “Advanced Explicit Congestion Notification”
- Other approaches around INT use ...
 - Custom protocols
 - Non-standard protocol modifications
- Presented at last year’s SIGCOMM: **TCP-INT**

TCP-INT: Delivering INT to Hosts

- Employs a custom **TCP header option**
 - Standard protocol functionality
- Provides telemetry from most congested switch
 - Fixed-size overhead
- Aimed at P4-programmable (Tofino) switches



Grzegorz Jereczek, Theo Jepsen et al.

“TCP-INT: Lightweight Network Telemetry with TCP Transport”.

In: *Proceedings of the SIGCOMM '22 Poster and Demo Sessions*.

Challenge: Integrating INT and Kernel TCP

- No kernel interface for custom IP or TCP options

Challenge: Integrating INT and Kernel TCP

- No kernel interface for custom IP or TCP options
- Existing INT approaches are hard to deploy, requiring ...
 - Kernel modifications
 - Kernel bypass (dpdk, ...)
 - Specialized NIC hardware

Challenge: Integrating INT and Kernel TCP


- No **traditional** kernel interface for custom IP or TCP options
- Existing INT approaches are hard to deploy, requiring ...
 - Kernel modifications
 - Kernel bypass (dpdk, ...)
 - Specialized NIC hardware
- We use eBPF!
 - SOCK_OPS programs can handle any TCP options



eBPF for Telemetry-Powered Congestion Control

Applying our solutions:

- A novel implementation of CC + INT
 - Enabled by eBPF
- A realization of the PowerTCP algorithm for Linux
 - Built on network telemetry
 - Targeted at datacenter networks
- Achieves high throughput and low delay

 **Vamsi Addanki, Oliver Michel, Stefan Schmid.**
“PowerTCP: Pushing the Performance Limits of Datacenter Networks”.
In: 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22).

Congestion Control in eBPF

- Kernel interface for CC algorithms:

```
struct tcp_congestion_ops {  
    void (*cong_control)(struct sock *sk, const struct rate_sample *rs);  
    void (*cwnd_event)(struct sock *sk, enum tcp_ca_event ev);  
    u32 (*undo_cwnd)(struct sock *sk);  
    ...  
};
```

- Usable in eBPF with STRUCT_OPS program type

Congestion Control in eBPF

- Kernel interface for CC algorithms:

```
struct tcp_congestion_ops {  
    void (*cong_control)(struct sock *sk, const struct rate_sample *rs);  
    void (*cwnd_event)(struct sock *sk, enum tcp_ca_event ev);  
    u32  (*undo_cwnd)(struct sock *sk);  
    ...  
};
```

- Usable in eBPF with STRUCT_OPS program type
- Few implementation differences/challenges:
 - No signed division in eBPF
 - Different way to store state: maps

CC in eBPF: Kernel Patches

Advanced CC: Assigns a socket pacing rate for optimal CWND usage

- Previously not allowed from eBPF
- We submitted patches, including other minor fixes
- Included since Linux 6.0

CC in eBPF: Kernel Improvements

Further eBPF improvements benefiting INT or CC:

- We submitted patches but still WIP:
 - Access to HW timestamps
- Wishlist:
 - Read and write access to IP options

Evaluation Questions

- Does the INT actually benefit the CC (queueing, stability, fairness)?
- Does the INT cause any unwanted side effects?

Evaluation Questions

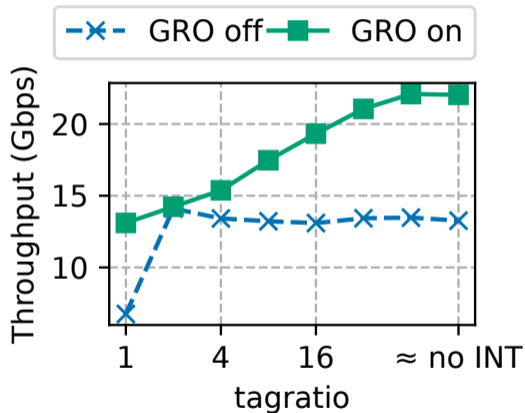
- Does the INT actually benefit the CC (queueing, stability, fairness)?
 - Yes!
- Does the INT cause any unwanted side effects?

Evaluation Questions

- Does the INT actually benefit the CC (queueing, stability, fairness)?
 - Yes!
- Does the INT cause any unwanted side effects?
 - **Unfortunately also yes**

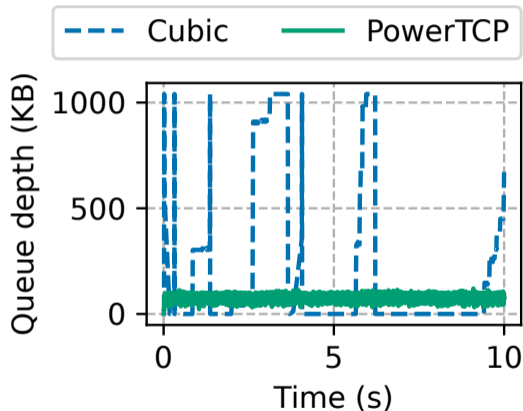
Employing INT: It breaks GRO

- GRO (generic receive offload) broken by frequent differences in TCP header
- Compensated by increasing MTU



INT-Powered CC: Queueing

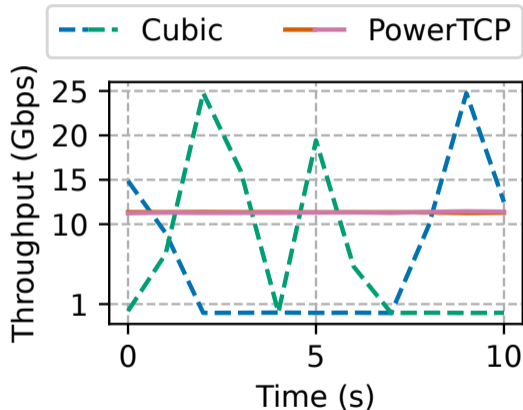
- PowerTCP maintains a low queue depth



INT-Powered CC: Fairness & Throughput

With two parallel flows received on a single link:

- PowerTCP maintains a stable, high, and fairly shared throughput



Conclusion

TCP's Third Eye:
Leveraging eBPF for Telemetry-Powered Congestion Control

Jörn-Thorben Hinz



- INT: Great new opportunities for congestion control
- eBPF: Enables INT usage and deployment **now**

Our code is available online:

<https://github.com/inet-tub/powertcp-linux>