eBPF Based Container Networking

A Network Performance Comparison

Nick de Bruijn

July 4, 2017

University of Amsterdam

Introduction

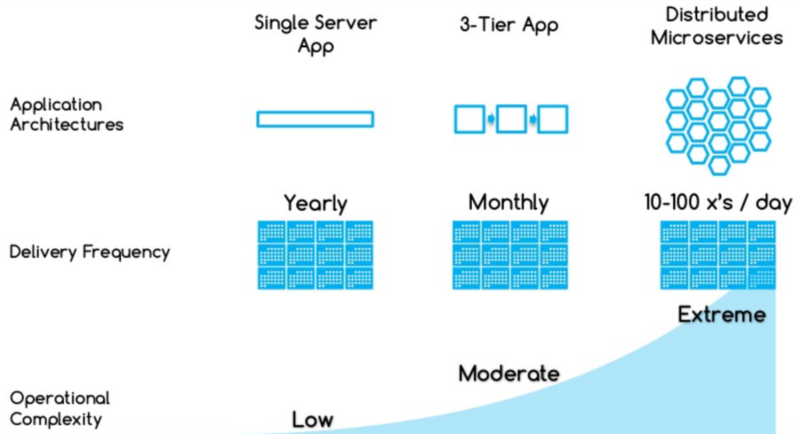


Figure 1: Microservices and Containers[1]

[1] https://www.slideshare.net/Docker/cilium-network-and-application-security-with-bpf-and-xdp-thomas-graf-covalent-io

Introduction - Iptables

Iptables:

- $ iptables -A INPUT -p tcp -s 10.0.0.23 –dport 80 -m conntrack –ctstate NEW -j ACCEPT

Research Goal

Research goal:

- Evaluate the usability of Cilium as a packet filtering system in a container (Microservices) infrastructure.

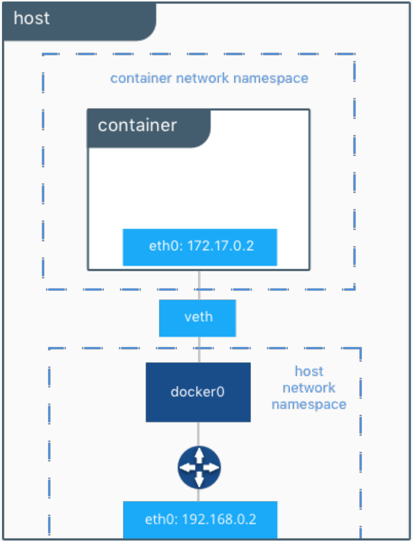
Research Questions

- What throughput and latency we get in the case of using Cilium's eBPF program and Linux's Iptables as packet filter?
- What effect does the number of security policies have on the throughput and latency in both cases?
- Is there a turn point in performance when increasing the number of security policies?

Background

Docker Networking

- Endpoints (Container eth0)
- Virtual Ethernet devices (veth)
- Bridge on the host (docker0)



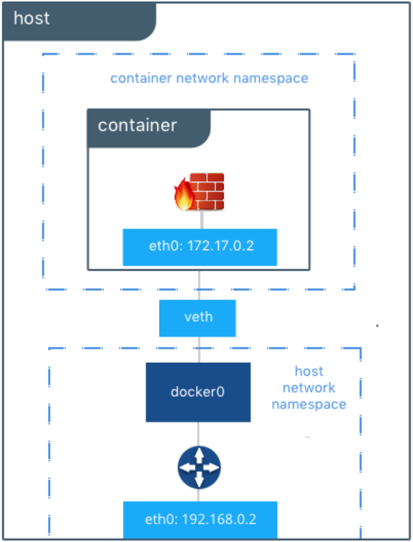
[1] Figure: https://success.docker.com/Architecture/Docker$_R$*eference*$_A$*rchitecture*

Docker Networking - Communication

- Endpoints (Container eth0)
- Virtual Ethernet devices (veth)
- Bridge on the host (docker0)

Packet filtering:

- On container



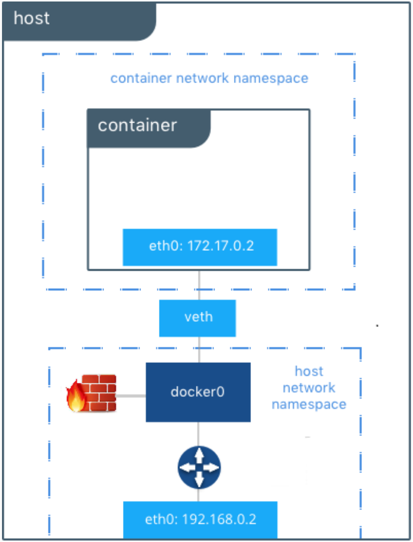
Docker Networking - Communication

Components:

- Endpoints (Container eth0)
- Virtual Ethernet devices (veth)
- Bridge on the host (docker0)

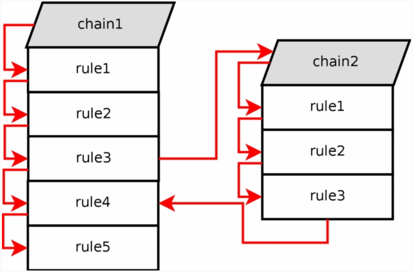
Packet filtering:

- On container
- On the bridge



Iptables - Performance penalty?

- Uses chains with rules
- Each chain contains 0 or more rules
- Top down approach
- Checks until match is found
- So placement is important



[2]

[2] Figure: http://www.iptables.info/en/structure-of-iptables.html

What is Cilium?

- Opensource project
- Adds a layer on top of the existing container environment (Docker)
- To improve container networking and policy enforcement
- No Iptables / bridges
- Relies on eBPF programs

What is eBPF (extended Berkeley Packet Filter)?

eBPF is used to extend the functionality of the kernel at runtime.

- It's effectively a small kernel based machine
    - 10 64bit registers
    - 512 byte stack
    - Data structures are known as maps
- Has a verifier to ensure the program is safe
    - No loops, max 4k instructions, no more then 64 maps.

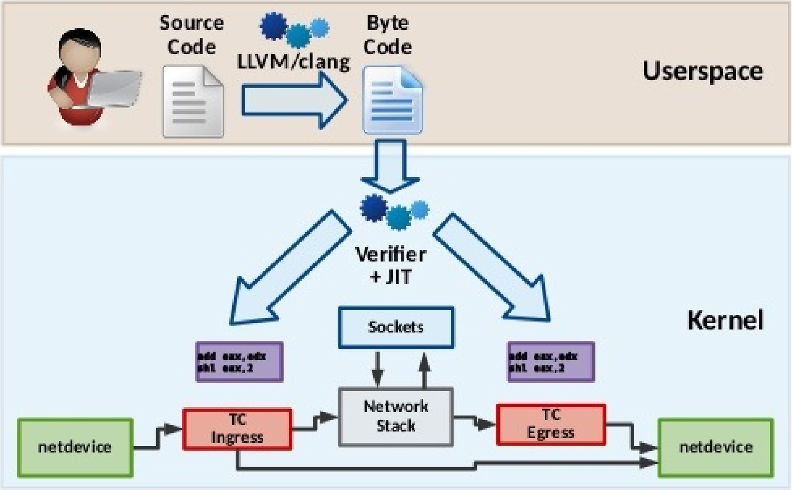


Figure 2: eBPF Overview[3]

[3] https://www.slideshare.net/Docker/cilium-bpf-xdp-for-containers-66969823

extended Berkley Packet Filter - Functionality

1. Rewrite packet content
2. Extend/trim packet size
3. Redirect to other netdevices
4. Enforce policies
5. On the fly program generation

Cilium - Network with eBPF

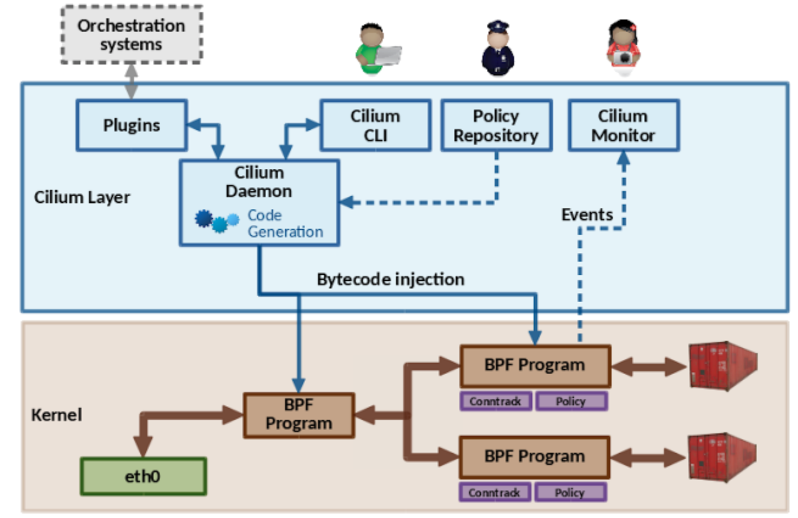


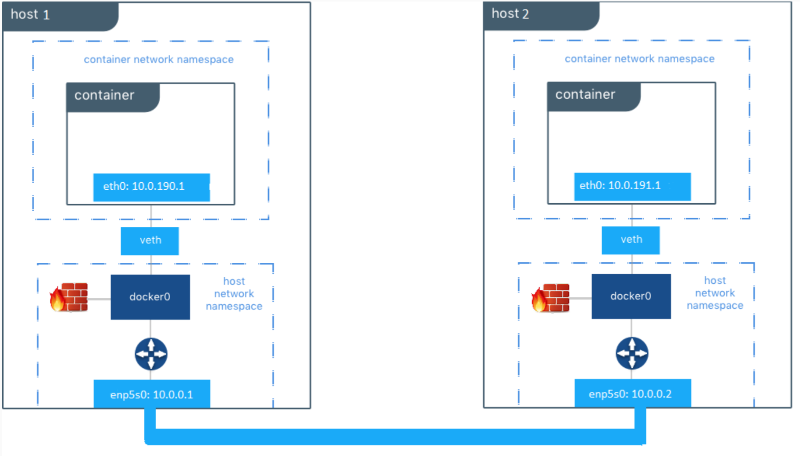
Figure 3: eBPF with Cilium[4]

[4] https://www.slideshare.net/Docker/cilium-bpf-xdp-for-containers-66969823

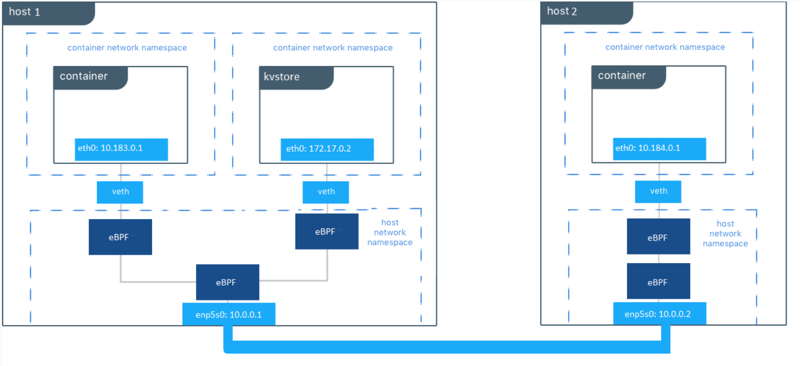
Figure 4: Cilium Policy Using Json

Approach

Approach - Docker environment



Approach - Cilium environment



Approach - Scenario

Performed tests on two scenarios:

- Localhost
- And Multi-host

For each scenario we are interested in:

- The throughput and latency with no additional policies/rules.
- The change in performance whenever we start to increase the number of policies/rules.

Approach - Experiments

- Using Iperf3 to send a TCP_STREAM
- Using Netperf to send a TCP_RR (Request Response)

- Every test runs 1 minute. Every test is performed 10 times to determine the variation
- Every test runs with 0, 1, 5, 10, 25, 50, 100, and 200 policies

Results

Results - Throughput Localhost

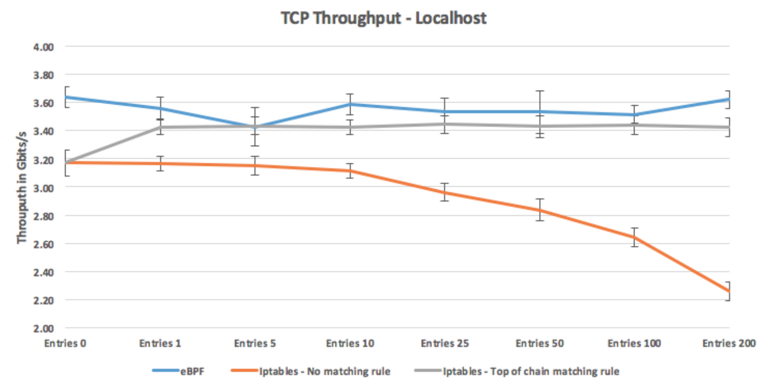


Figure 5: Throughput - localhost (Higher is better)

- Cilium's eBPF approach outperforms the IPtable approach.
- Number of Cilium policies does not affect the throughput
- Number of no matching Iptables rules greatly affect the throughput

Results - Latency Localhost

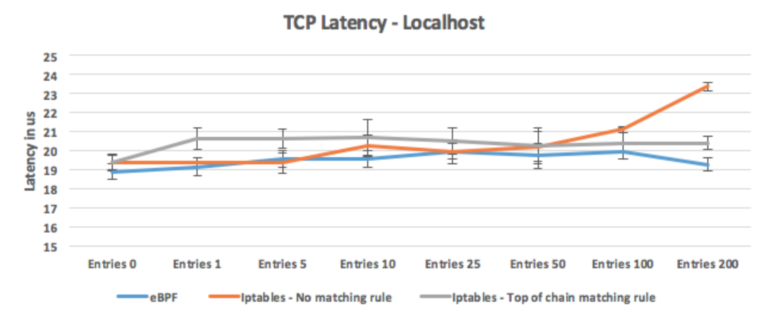


Figure 6: TCP Latency - localhost (Lower is better)

- Same observation as the throughput
- Cilium's eBPF approach has a lower latency

Results - Throughput Remote Containers

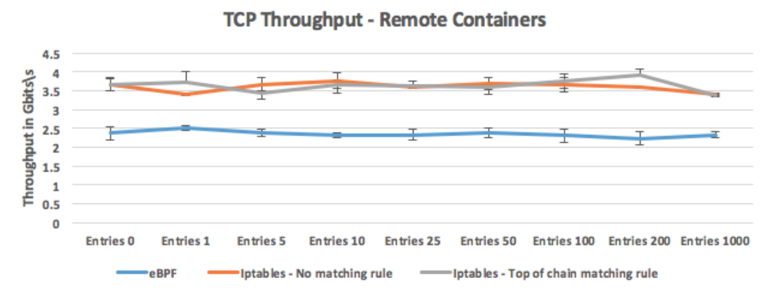


Figure 7: TCP Throughput - Remote Host (Higher is better)

- Different observation than on Localhost
- Cilium's eBPF seems to perform less
- Iptables show no performs penalty until 1000 policies

Results - Latency Remote Containers

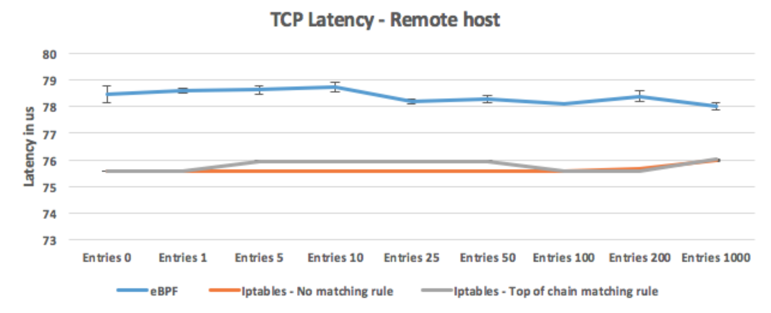


Figure 8: TCP Latency - Remote Host (Lower is better)

- Same observation as the remote throughput
- Cilium's eBPF approach has a higher latency

Conclusion

Conclusion

Overal:

1. Cilium seems like a promising project.
2. We can define L3, L4, and L7 policies

Performance wise:

1. The performance is not influenced by number of policies.
2. Cilium shows to perform better in the situation of local containers.
3. Room for improvements for multi-host enviornments

Open issues & Future work

- Test the VXLAN overlay overhead used by Docker and Cilium
- Do Kernel traces to get a better understanding of which path packets take in the kernel.
- Optimize both approaches to see what the best possible throughput and latency can be reached for each approach.
- Test Cilium using XDP to offload the system.

Thank you for your attention,

Questions?