CONTAINER NETWORKING SOLUTIONS

Anjali Singhai Jain, Nupur Jain, Intel

Notices & Disclaimers

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.

Your costs and results may vary.

Intel technologies may require enabled hardware, software or service activation.

Agenda

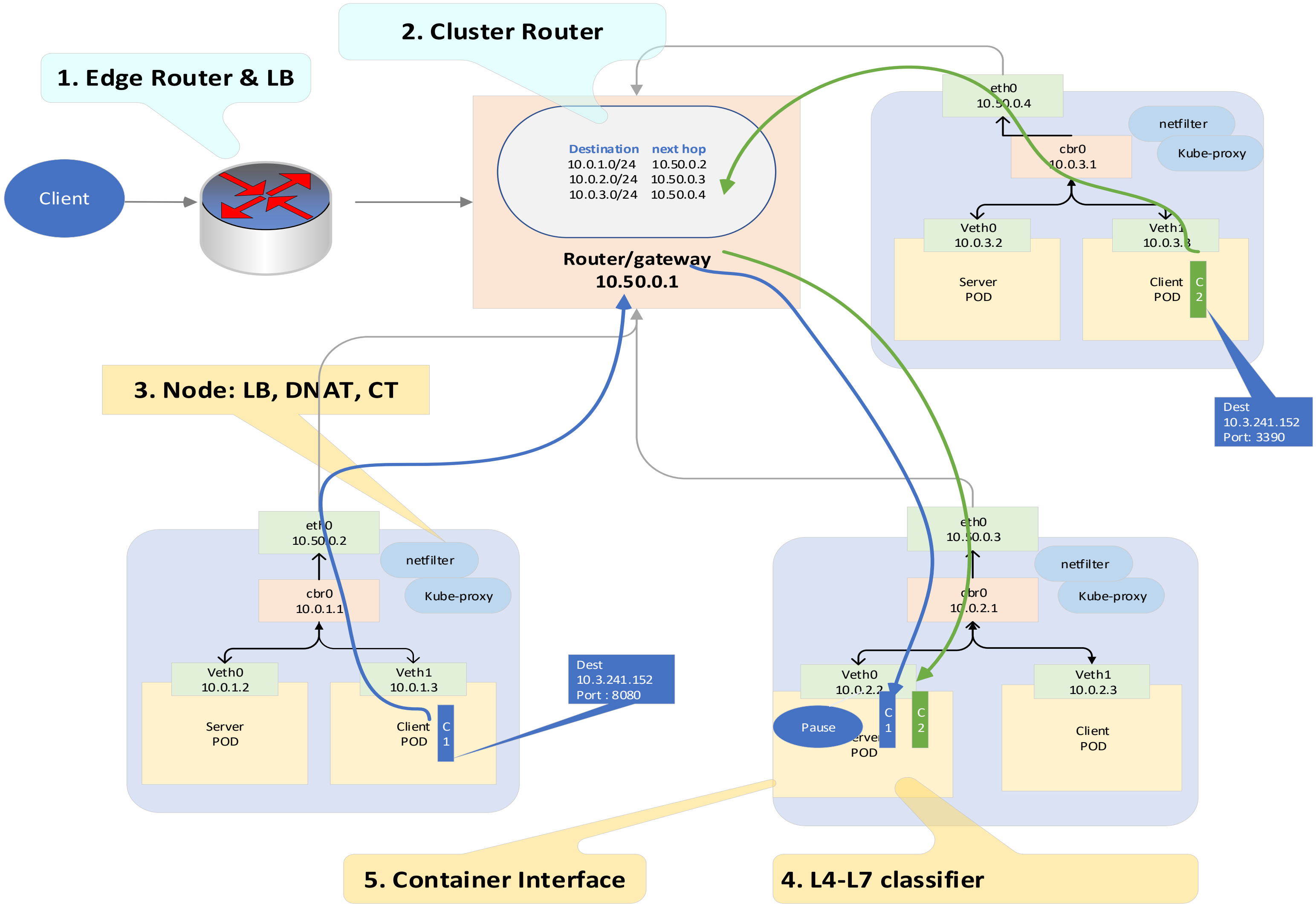
Introduction

- Container Network Functions & Interfaces
- Limitations
- Container Interface Classifier
- Community Solutions
- Our Proposed Solution
- Consolidation

Functionality Offload

- P4 Sample -Kubeproxy
- P4 Connection tracking
- P4 L4-L7 Classifier

Container Networking



Current State of Deployment

Need for Scale
CSP Container Scale Needed ~10K

+

Device performance
Throughput, Programmability, Resource Scaling has increased significantly

+

Effective utilization
of the system resources (cores, memory, network) to improve performance and packing of containers

=

Ideal Container Network Solution

5. Existing interfaces for containers

Shared (Pod Interfaces)

- Stacked netdevs on a PCIe PF netdev
- Assigned to container namespaces
- Examples : MACVLAN, IPVLAN, bridge

Dedicated (Pod Interfaces)

- SR-IOV VFs
- Too heavy
  - Separate PCIE config space
  - HW based packet replication for broadcast, multicast – higher PCI BW utilization

Why Hardware Accelerate ?

End to end Maximize throughput

- Avoid the SW long path which limits how much a Server Pod can handle.

End to end Native Scale out

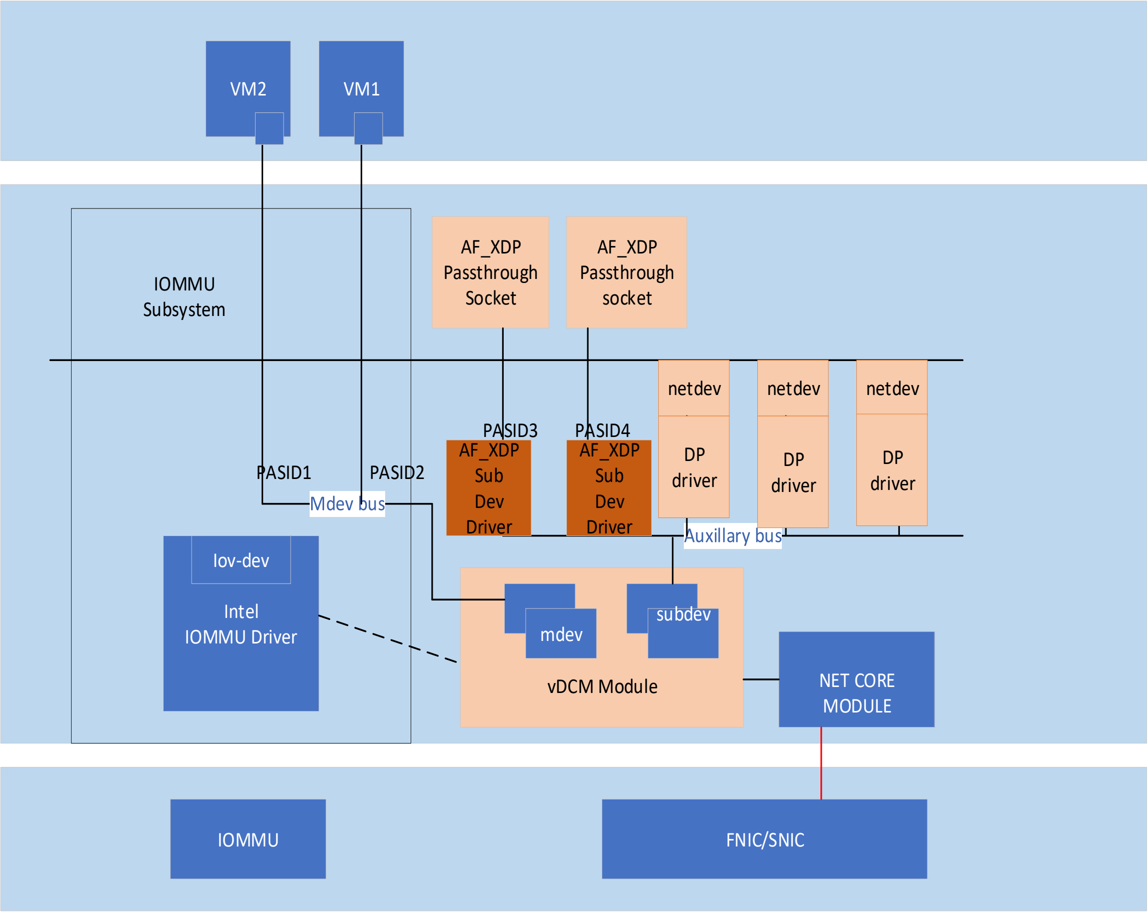
- By Reducing Latency and Jitter introduced by kernel to user context switch in present AF_XDP memory model.
- Dedicated resources takes away the need for OS to schedule on a shared resource. OS overhead for managing resources is gone.
  - CPU scheduling, memory management etc.

Security & Isolation

- Queue level isolation.

Assignable Container interfaces using X-IOV (S-IOV) & User Interrupts

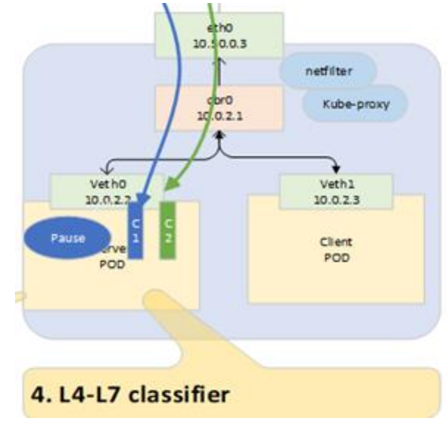
- Hardware Assisted Virtualization
- Highly scalable and high- performance sharing of I/O devices across isolated domains
- Assignable device Interfaces =>User Container Interfaces
- Platform Scalability using PASID
- Support Virtual Device Composition
- *RID and *PASID identifies the address space associated with the request
- *ADI Memory Mapped regions



*RID = Requester ID
*PASID = Process Address Space Identifier
*ADI Assignable Device interfaces
SIOV Spec : Reference Number: 337679-001, Revision: 1.0

4. Container Interface Classifier: Solution

- L4-L7 Classifier and forwarder in the HW
- This extends the HW Offloaded vSwitch Classification End Point to the Container Interface.



Option1: HW Offload the classifier & forwarder

- AF_XDP raw_socket bound to a HW vPort/QP through Side band filters.
- Provide inline filters to be added in HW as part of TX packet.
  - ATR style in ADQ

Option2: HW Offload the Classifier

- Provide a meta data classification hint to kernel/user with a packet.
  - Flow mark or a 32bit hash value based on L4-L7 fields.

3. Node: Load Balancer, DNAT, CT

Existing Solutions

Kube-proxy

- Kernel Netfilters - Not performant
  - Iptables O(n) chains proportional to size of cluster, in-place rule modifications not possible.
  - IPVS O(1) - hash ipset but do not work well with other services requiring iptables for filtering
- Kernel with eBF/XDP - Accelerated

Connection Tracking

- Robust to syn floods but limited by max size
- No flexibility, fixed hash algorithm and field selection for hash

Kernel Overall not very flexible, latencies due to irq processing, context switching, slow API configuration interfaces

Community's Approach - eBPF/XDP

Benefits - Performant than the kernel

Designed as an alternative to DPDK.

Flexibility, code injected into the kernel

Ability to reload programs on-the-fly

Network Functions - Network Policy, Encryption, Load Balancer, Firewall, Monitoring etc

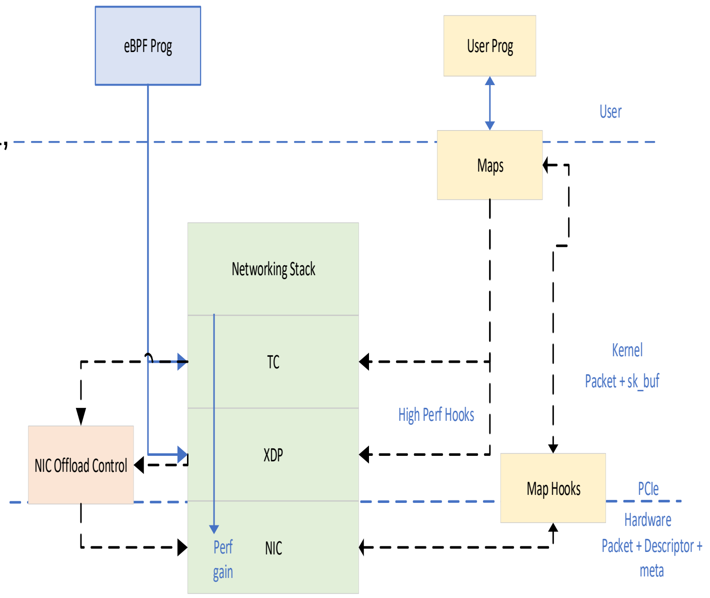
Plugins - Cilium, Callico, Katran (facebook), etc

Limitations -

- Not HW offloadable to ASICs.
  - Note: Netronome uses NPU general Purpose cores
- More cores required to scale connections
- Kernel/user space context switching
- General purpose CPUs and Memory architecture is not ideal for Deep table Lookup

Solution –

- Purpose built ASICs, purpose -built cores and dedicated Context aware caches may be the way to go



eBPF Implementation – Cilium

- Cilium does connect-time load balancing by hooking into the kernel XDP/TC hook on the receive.
- When a program tries to connect to a Kubernetes service, Cilium intercepts the connection attempt, load balances with DNAT's to directly connect to the backend pod's IP instead.

Throughput in queries/s – test run with fortio and ngnix. 2 clients, 2 NodePort IPs, 2 backend Pods per NodePort

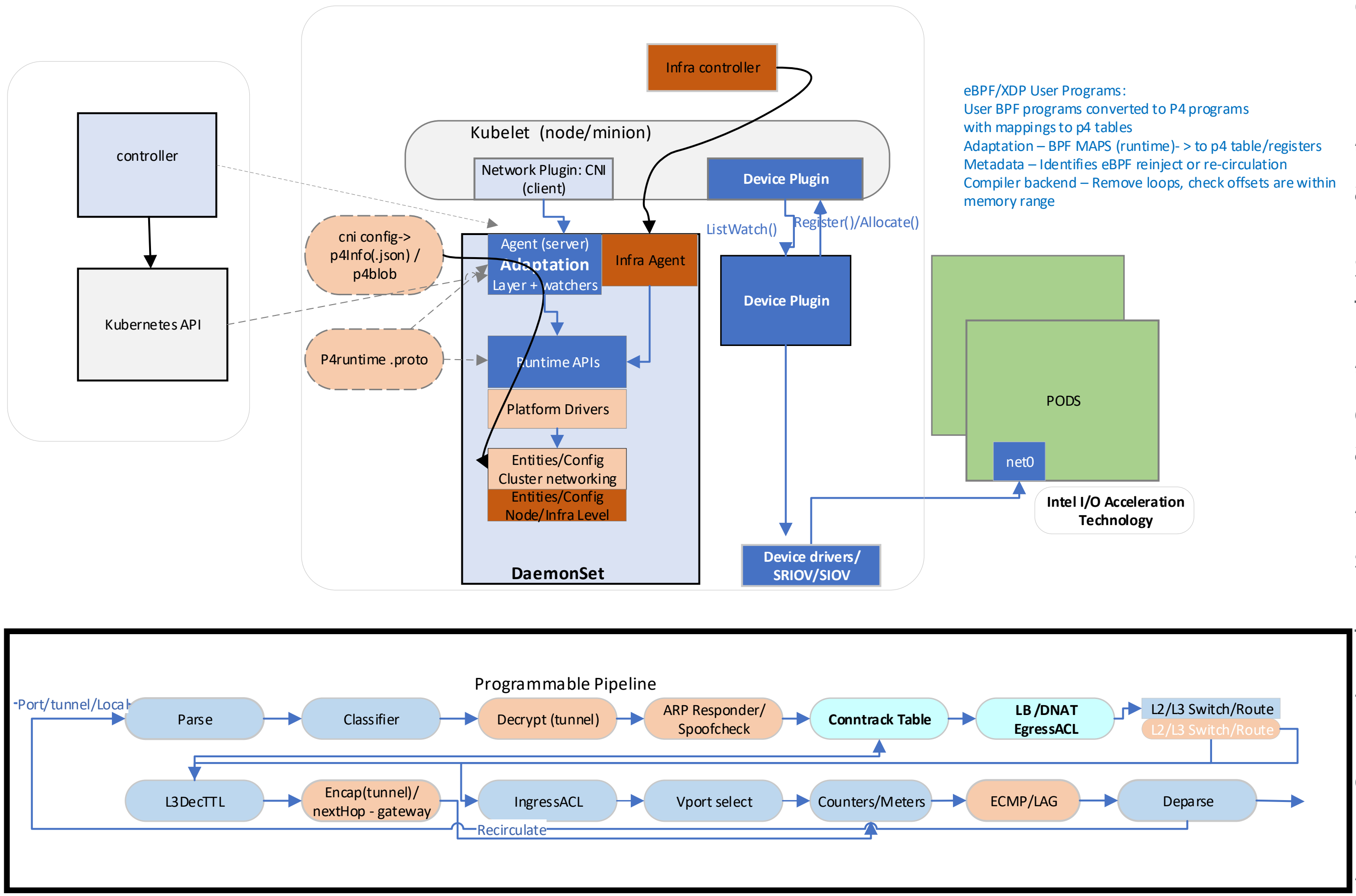
No XDP, 2 CLV interfaces			
Fortio + Nginx	Client 1	Client 2	Total
100B	47578.6	50411.5	97990.1
1500B	46174.4	49990.4	96164.8

XDP, 1 CLV interface			
Fortio + Nginx	Client 1	Client 2	Total
100B	64038.7	53970	118008.7
1500B	49623.5	48369.7	97993.2

No-XDP and XDP performed similar with 100,000+ connections. CPU consumption in both reaches to >8 cores with more sessions

We suspect, XDP path based on this experiment, does not take advantage of hardware XDP_REDIRECT queue designed to send packets to another interface, hence no significant performance gains

Our Approach – The Whole Datapath



*Programmable MAT tables. Contract between control plane and data plane for runtime control

*Device capability defined by architecture. Eg. wildcard match -TCAM, Exact match-SRAM. Compiler responsible for mapping.

*Parallel lookups, conditional actions & atomicity

*Features are defined in the software. Faster introduction, verification, test and deployment.

*Programmer defined 1) parse graph; headers and orders. 2) Packet modifications

*Counters, meters, stateful registers, hash functions, ALU, TTLs, PRE

Node LB Data Plane P4

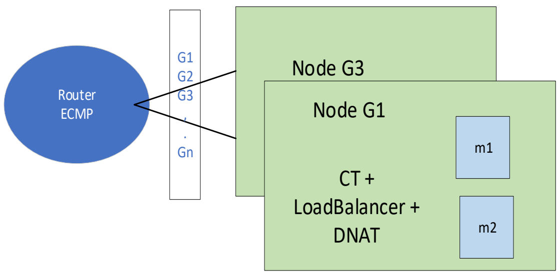


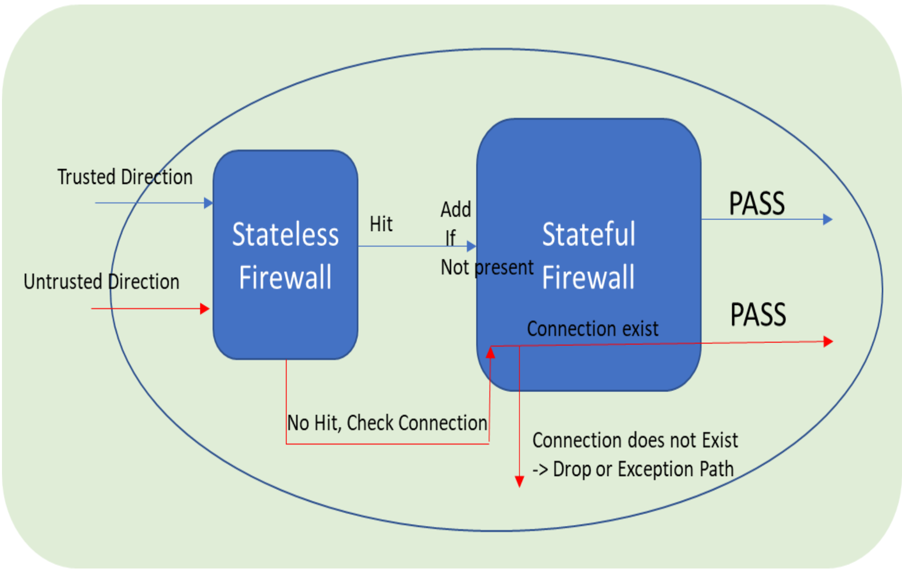
Table Entry	Key (ClusterIP, port	Member group
t1	10.3.241.152	g1
t2	10.3.241.159	g2
t3	10.3.241.170	g3

Group Re	Members
g1	m1, m2,m3
g2	m1,m2
g3	m1,m2

Member ref	Action Spec
m1	10.0.1.2, 8080, 02:42:88:7e:17:61
m2	10.0.2.2,8080, f6:27:14:ce:4a:7d

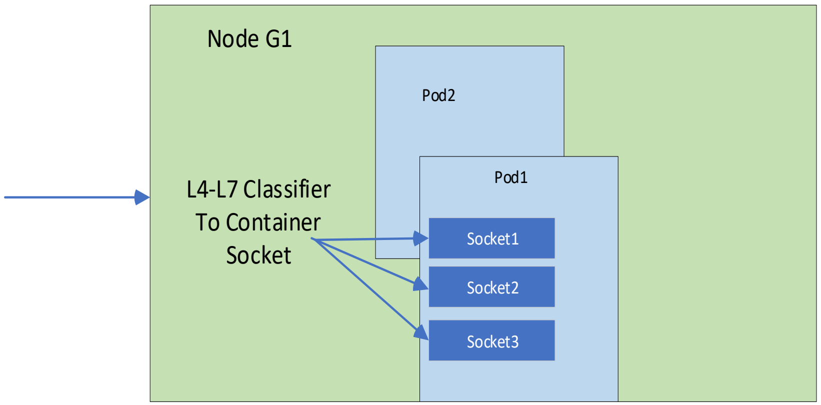
```
// Proxy LB on cluster service IP to an endpoint on a POD
extern ActionSelector{
        /// Construct a selection table for a given ActionProfile.
        ActionSelector(ActionProfile action_profile,
                        Hash<_> hash,
                        SelectorMode_t mode,
                        bit<32> max_group_size,
                        bit<32> num_groups);

        ActionSelector(bit<32> size, Hash<_> hash, SelectorMode_t mode);
}


control simple_lb(inout headers hdr,
        inout metadata meta,
        switch_uint32_t lb_table_size,
        inout standard_metadata_t standard_metadata) {

        //Chose an extern hash or PNA hash
        Hash<switch_uint32_t>(HashAlgorithm_t.CRC32) selector_hash;

        ActionSelector(
                1024, selector_hash, SelectorMode_t.FAIR) pod_selector;

        // Pick an entry and apply DNAT
        action set_nhop(bit<48> pod_dmac, bit<32> pod_ipv4, bit<9> port) {
                hdr.ethernet.dstAddr = pod_dmac;
                hdr.ipv4.dstAddr = pod_ipv4;
                hdr.tcp.dstPort = port;
                standard_metadata.egress_spec = port;
        }

        // hash to use 5-tuple
        // can be tcp, udp, sctp
        table lb {
                key = {
                        hdr.ipv4.dstAddr : exact;
                        hdr.tcp.dstPort : exact;
                        hdr.ipv4.srcAddr : selector;
                        hdr.ipv4.dstAddr : selector;
                        hdr.ipv4.tcp.srcPort : selector;
                        hdr.ipv4.tcp.dstPort : selector;
                        hdr.ipv4.protocol : selector
                }

                actions = {
                        NoAction;
```

CT P4 Data Plane




```
action ct_aging_table_hit (
    @per_entry_state in    FlowId_t flow_id,
    @per_entry_state inout ExpireTimeSelection_t expire_time)
{
    // my_flow_id = flow_id;
    if (modify_expire_soon_on_hit) {
        expire_time = new_expire_time_selection;  // timers per state
        modify_entry_to_expire_soon();
    } else if (update_expire_time) {
        expire_time = new_expire_time_selection;
        restart_expire_timer();            // sweep_count = 0
    } else {
        restart_expire_timer();            // sweep_count = 0
    }
}

action ct_aging_table_miss() {
    FlowId_t my_flow_id;
    bool add_succeeded;
    if (add_on_miss) {
        my_flow_id = allocate_flow_id();
        add_succeeded =
            add_entry("ct_aging_table_hit",  // name of action
                      (ct_aging_table_hit_params_t)
                      {flow_id = my_flow_id,
               expire_time = new_expire_time_selection});
        // add_entry() initializes the new entry as if
        // restart_expire_timer() had been called on it,
    }
}

// CT 5 Tuple table
table ct_aging_table {
    key = {
        // P4 developer gets to select the key fields they want,
        // e.g. This is a 5 tuple plus a field like zone id , used to guarantee
        // that entries in different IP private address domains are
        // unique.
    }
    actions = {
        // @tableonly and @defaultonly are standard annotations
        // defined in the P4_16 language specification.
        @tableonly   ct_aging_table_hit;
        @defaultonly ct_aging_table_miss;
    }
    // This is a straw man syntax.  Probably want more than 'true' so
    // that this code might contain other compile-time options.
    add_on_miss = true;
    idle_timeout = true;
    const default_action = ct_aging_table_miss;
}

apply {
```

L4-L7 P4 Classifier




```
control L4_L7_classifier(inout headers hdr,
        inout metadata meta,
        switch_uint32_t lb_table_size,
        inout standard_metadata_t standard_metadata) {

        // Forward to a Container Interface
        action set_queue(bit<16> queue_id) {
            standard_metadata.egress_spec = queue_id;
        }

        // Set a flow ID meta data
        action set_flow_id(bit<32> mark) {
            standard_metadata.flow_id = mark;
        }

        // exact match using n-tuple
        // can be tcp, udp, sctp
        table lb {
            key = {
                hdr.ipv4.dstAddr : exact;
                hdr.udp.dstPort : exact;
                hdr.udp.srcPort : exact;
                hdr.ipv4.srcAddr : exact;
                hdr.udp.quic.cid : exact
            }

            actions = {
                NoAction;
                set_queue;
                flow_count.count();
                set_flow_id;
              }

            const default_action = Drop;
            counters = flow_counter;
            size = lb_table_size; // can be configured
            implementation = pod_selector;
        }
}
```

Opens

- Not every eBPF program can be HW offloaded as is. We are looking at all use cases.
- We would like to get community support in converting some well defined XDP implementations to p4 programs.
- P4 extensions or externs is an option for complete packet transformations like Crypto, checksum, packet replication etc.

Conclusion:

Components for Native Scale out of Container Networking

Enhanced SW Container & Pod Interfaces

Standard Container Control Plane SW

SW support for Platform Light-Weight Virtualization

SW to support FNIC/SNIC HW to provide Container Networking offload Through Programmable DP

Components	
Standard (No Change)	(blue)
Need SW Enhancements to benefit from HW	(green)

Contacts

P4 Code will be on github soon…

Please email for more info…

Contacts: Anjali Singhai Jain anjali.singhai@intel.com

Nupur Jain nupur.jain@intel.com

Intel Container Networking Team:

Anjali Singhai Jain, Nupur Jain, Amritha Nambiar, Pawel Szymanski, Shaopeng He, Phani Burra,	Dan Daly, Yadong Li, Sridhar Samudrala, Kiran Patil, Liang Cunming , Edwin Verplanke

eBPF Implementation Characterization – Cilium

Stack trace (NO XDP)

vmlinux	615.317s
nginx	55.704s
[Unknown]	24.101s
[Outside any known module]	24.101s
cls_bpf_classify	21.866s
↖ __tcf_classify ← tcf_classify_in	1.173s
↖ __tcf_classify ← tcf_classify ←	1.042s
[Unknown]	0.020s
↖ func@0x7a5bd4 ← func@0x79	0s
func@0x7a0570	0s
func@0x3527ee1	0s
ice.ko	23.259s
ip_tables.ko	20.728s
libc-2.13.so	11.371s
amplxe-perf	8.445s
cls_bpf.ko	8.339s
containerd-shim	8.209s

Stack trace (XDP)

[Unknown]	28.626s	52,704,500,000
[Outside any known modu	28.626s	52,704,500,000
cls_bpf_classify	17.400s	29,923,000,000
↖ bpf_prog_run_xdp ← i	8.194s	21,148,500,000
↖ __tcf_classify ← tcf_cl	1.463s	609,500,000
↖ __tcf_classify ← tcf_cl	1.273s	621,000,000
↖ ice_napi_poll ← napi_	0.236s	322,000,000
[Unknown]	0.035s	23,000,000
func@0x79fe60	0.015s	11,500,000
↖ func@0x7a19d5 ← fur	0.005s	0
func@0x3527ee1	0.005s	34,500,000
↖ func@0x7a0570 ← fur	0s	11,500,000
ice.ko	25.519s	26,553,500,000
ip_tables.ko	17.896s	32,798,000,000
libc-2.13.so	11.096s	22,482,500,000
containerd-shim	7.943s	12,788,000,000
cls_bpf.ko	7.367s	8,533,000,000

- XDP vs No XDP, CPU utilization is quite similar
- Benefits of XDP is being able to bypass the kernel stack in case of redirect.
- Redirect to external port requires dedicate HW Redirect TX queue.
- XDP Benefits can be derived from dedicated HW resources.

eBPF Implementation Characterization – Cilium - cont

Cilium chains in iptables

```
# Generated by iptables-save v1.6.1 on Tue Dec  1 13:22:26 2020
*raw
:PREROUTING ACCEPT [339745057:40541787768]
:OUTPUT ACCEPT [1966765:7841806195]
:CILIUM_OUTPUT_raw - [0:0]
:CILIUM_PRE_raw - [0:0]
-A PREROUTING -m comment --comment "cilium-feeder: CILIUM_PRE_raw" -j CILIUM_PRE_raw
-A OUTPUT -m comment --comment "cilium-feeder: CILIUM_OUTPUT_raw" -j CILIUM_OUTPUT_raw
-A CILIUM_OUTPUT_raw -o lxc+ -m mark --mark 0xa00/0xfffffeff -m comment --comment "cilium: NOTRACK for proxy return traff
-A CILIUM_OUTPUT_raw -o cilium_host -m mark --mark 0xa00/0xfffffeff -m comment --comment "cilium: NOTRACK for proxy retur
-A CILIUM_PRE_raw -m mark --mark 0x200/0xf00 -m comment --comment "cilium: NOTRACK for proxy traffic" -j NOTRACK
COMMIT
# Completed on Tue Dec  1 13:22:26 2020
# Generated by iptables-save v1.6.1 on Tue Dec  1 13:22:26 2020
*mangle
:PREROUTING ACCEPT [339745170:40541800199]
:INPUT ACCEPT [1689410:7607891639]
:FORWARD ACCEPT [338055760:32933908560]
:OUTPUT ACCEPT [1966765:7841806195]
:POSTROUTING ACCEPT [340022530:40775715026]
:CILIUM_POST_mangle - [0:0]
:CILIUM_PRE_mangle - [0:0]
-A PREROUTING -m comment --comment "cilium-feeder: CILIUM_PRE_mangle" -j CILIUM_PRE_mangle
-A POSTROUTING -m comment --comment "cilium-feeder: CILIUM_POST_mangle" -j CILIUM_POST_mangle
-A CILIUM_PRE_mangle -m socket --transparent -m comment --comment "cilium: any->pod redirect proxied traffic to host prox
-A CILIUM_PRE_mangle -p tcp -m mark --mark 0xd7ae0200 -m comment --comment "cilium: TPROXY to host cilium-dns-egress prox
-A CILIUM_PRE_mangle -p udp -m mark --mark 0xd7ae0200 -m comment --comment "cilium: TPROXY to host cilium-dns-egress prox
COMMIT
# Completed on Tue Dec  1 13:22:26 2020
# Generated by iptables-save v1.6.1 on Tue Dec  1 13:22:26 2020
*filter
:INPUT ACCEPT [1689410:7607891639]
:FORWARD DROP [0:0]
:OUTPUT ACCEPT [1966765:7841806195]
:CILIUM_FORWARD - [0:0]
:CILIUM_INPUT - [0:0]
:CILIUM_OUTPUT - [0:0]
-A INPUT -m comment --comment "cilium-feeder: CILIUM_INPUT" -j CILIUM_INPUT
-A FORWARD -m comment --comment "cilium-feeder: CILIUM_FORWARD" -j CILIUM_FORWARD
-A OUTPUT -m comment --comment "cilium-feeder: CILIUM_OUTPUT" -j CILIUM_OUTPUT
-A CILIUM_FORWARD -o cilium_host -m comment --comment "cilium: any->cluster on cilium_host forward accept" -j ACCEPT
-A CILIUM_FORWARD -i cilium_host -m comment --comment "cilium: cluster->any on cilium_host forward accept (nodeport)" -j
-A CILIUM_FORWARD -i lxc+ -m comment --comment "cilium: cluster->any on lxc+ forward accept" -j ACCEPT
-A CILIUM_FORWARD -i cilium_net -m comment --comment "cilium: cluster->any on cilium_net forward accept (nodeport)" -j AC
-A CILIUM_INPUT -m mark --mark 0x200/0xf00 -m comment --comment "cilium: ACCEPT for proxy traffic" -j ACCEPT
-A CILIUM_OUTPUT -m mark --mark 0xa00/0xfffffeff -m comment --comment "cilium: ACCEPT for proxy return traffic" -j ACCEPT
-A CILIUM_OUTPUT -m mark ! --mark 0xe00/0xf00 -m mark ! --mark 0xd00/0xf00 -m mark ! --mark 0xa00/0xe00 -m comment --comm
COMMIT
# Completed on Tue Dec  1 13:22:26 2020
# Generated by iptables-save v1.6.1 on Tue Dec  1 13:22:26 2020
*nat
:PREROUTING ACCEPT [0:0]
:INPUT ACCEPT [0:0]
```

Cilium's own conntrack table

```
TCP IN 12.91.212.200:37386 -> 10.0.0.241:80 expires=17272887 RxPackets=18796 RxBytes=2057656 RxFlagsSeen=0x1b LastRxReport=17251794 TxPackets=18915 TxBytes=4544990 TxFlagsSeen=
TCP OUT 13.91.212.200:33224 -> 10.0.0.55:80 expires=17275420 RxPackets=83205 RxBytes=19555155 RxFlagsSeen=0x1a LastRxReport=17254320 TxPackets=80142 TxBytes=9897097 TxFlagsSeen
TCP OUT 12.91.212.202:30007 -> 12.91.212.200:36342 service expires=17272760 RxPackets=0 RxBytes=6 RxFlagsSeen=0x00 LastRxReport=0 TxPackets=0 TxBytes=0 TxFlagsSeen=0x1f LastTxR
TCP OUT 12.91.212.202:38900 -> 12.91.212.200:30008 expires=17274846 RxPackets=2 RxBytes=120 RxFlagsSeen=0x14 LastRxReport=17253753 TxPackets=1 TxBytes=74 TxFlagsSeen=0x02 LastT
TCP OUT 12.91.212.200:43362 -> 10.0.0.246:80 expires=17275288 RxPackets=141178 RxBytes=33482100 RxFlagsSeen=0x1a LastRxReport=17254194 TxPackets=137700 TxBytes=17005280 TxFlags
TCP IN 13.91.212.200:49604 -> 10.0.0.55:80 expires=17271556 RxPackets=90047 RxBytes=9859771 RxFlagsSeen=0x1b LastRxReport=17250463 TxPackets=91583 TxBytes=21846155 TxFlagsSeen=
TCP IN 13.91.212.200:55058 -> 10.0.0.55:80 expires=17273171 RxPackets=52349 RxBytes=5757922 RxFlagsSeen=0x1b LastRxReport=17252078 TxPackets=53989 TxBytes=49437539 TxFlagsSeen=
TCP OUT 12.91.212.200:43004 -> 10.0.0.241:80 expires=17275006 RxPackets=1016 RxBytes=227571 RxFlagsSeen=0x1a LastRxReport=17253912 TxPackets=923 TxBytes=113493 TxFlagsSeen=0x1e
TCP IN 12.91.212.200:43512 -> 10.0.0.246:80 expires=17275288 RxPackets=140974 RxBytes=17409857 RxFlagsSeen=0x1e LastRxReport=17254195 TxPackets=145276 TxBytes=34327857 TxFlagsS
TCP OUT 13.91.212.202:48040 -> 13.91.212.200:30008 expires=17274814 RxPackets=2 RxBytes=120 RxFlagsSeen=0x14 LastRxReport=17253721 TxPackets=1 TxBytes=74 TxFlagsSeen=0x02 LastT
TCP IN 12.91.212.200:44096 -> 10.0.0.241:80 expires=17275421 RxPackets=102519 RxBytes=12660607 RxFlagsSeen=0x1e LastRxReport=17254328 TxPackets=104458 TxBytes=24884841 TxFlagsS
TCP IN 12.91.212.200:43468 -> 10.0.0.246:80 expires=17275288 RxPackets=130334 RxBytes=16095587 RxFlagsSeen=0x1e LastRxReport=17254195 TxPackets=133864 TxBytes=31706643 TxFlagsS
TCP OUT 12.91.212.202:30007 -> 12.91.212.200:42992 service expires=17275006 RxPackets=0 RxBytes=6 RxFlagsSeen=0x00 LastRxReport=0 TxPackets=0 TxBytes=0 TxFlagsSeen=0x1e LastTxR
TCP OUT 13.91.212.202:30008 -> 13.91.212.200:54258 service expires=17272887 RxPackets=0 RxBytes=9 RxFlagsSeen=0x00 LastRxReport=0 TxPackets=0 TxBytes=0 TxFlagsSeen=0x1b LastTxR
TCP OUT 12.91.212.202:54410 -> 12.91.212.200:30007 expires=17274731 RxPackets=2 RxBytes=120 RxFlagsSeen=0x14 LastRxReport=17253638 TxPackets=1 TxBytes=74 TxFlagsSeen=0x02 LastT
TCP OUT 12.91.212.202:54426 -> 12.91.212.200:30007 expires=17274731 RxPackets=2 RxBytes=120 RxFlagsSeen=0x14 LastRxReport=17253638 TxPackets=1 TxBytes=74 TxFlagsSeen=0x02 LastT
TCP OUT 12.91.212.202:30007 -> 12.91.212.200:35150 service expires=17272401 RxPackets=0 RxBytes=8 RxFlagsSeen=0x00 LastRxReport=0 TxPackets=0 TxBytes=0 TxFlagsSeen=0x1f LastTxR
TCP IN 12.91.212.200:42930 -> 10.0.0.246:80 expires=17275006 RxPackets=1174 RxBytes=144442 RxFlagsSeen=0x1e LastRxReport=17253913 TxPackets=1292 TxBytes=289662 TxFlagsSeen=0x1a
TCP IN 13.91.212.200:33946 -> 10.0.0.55:80 expires=17275768 RxPackets=24996 RxBytes=3086447 RxFlagsSeen=0x1a LastRxReport=17254672 TxPackets=25670 TxBytes=6078920 TxFlagsSeen=0
TCP OUT 12.91.212.202:30007 -> 12.91.212.200:38580 service expires=17273496 RxPackets=0 RxBytes=6 RxFlagsSeen=0x00 LastRxReport=0 TxPackets=0 TxBytes=0 TxFlagsSeen=0x1b LastTxR
TCP IN 13.91.212.200:34256 -> 10.0.0.55:80 expires=17275769 RxPackets=21454 RxBytes=2649240 RxFlagsSeen=0x1a LastRxReport=17254675 TxPackets=22597 TxBytes=5255183 TxFlagsSeen=0
TCP OUT 13.91.212.200:33150 -> 10.0.0.55:80 expires=17275420 RxPackets=80644 RxBytes=19005294 RxFlagsSeen=0x1a LastRxReport=17254316 TxPackets=77972 TxBytes=9629102 TxFlagsSeen
TCP OUT 12.91.212.202:30007 -> 12.91.212.200:44004 service expires=17275421 RxPackets=0 RxBytes=8 RxFlagsSeen=0x00 LastRxReport=0 TxPackets=0 TxBytes=0 TxFlagsSeen=0x1e LastTxR
TCP OUT 13.91.212.202:30008 -> 13.91.212.200:55654 service expires=17273480 RxPackets=0 RxBytes=9 RxFlagsSeen=0x00 LastRxReport=0 TxPackets=0 TxBytes=0 TxFlagsSeen=0x1b LastTxR
TCP OUT 12.91.212.202:55190 -> 12.91.212.200:30007 expires=17274765 RxPackets=2 RxBytes=120 RxFlagsSeen=0x14 LastRxReport=17253672 TxPackets=1 TxBytes=74 TxFlagsSeen=0x02 LastT
TCP OUT 12.91.212.202:39914 -> 12.91.212.200:30008 expires=17274929 RxPackets=2 RxBytes=120 RxFlagsSeen=0x14 LastRxReport=17253836 TxPackets=1 TxBytes=74 TxFlagsSeen=0x02 LastT
TCP OUT 12.91.212.202:38932 -> 12.91.212.200:30008 expires=17274846 RxPackets=2 RxBytes=120 RxFlagsSeen=0x14 LastRxReport=17253753 TxPackets=1 TxBytes=74 TxFlagsSeen=0x02 LastT
TCP OUT 12.91.212.202:38888 -> 12.91.212.200:30008 expires=17274846 RxPackets=2 RxBytes=120 RxFlagsSeen=0x14 LastRxReport=17253753 TxPackets=1 TxBytes=74 TxFlagsSeen=0x02 LastT
TCP OUT 12.91.212.202:30007 -> 12.91.212.200:44106 service expires=17275421 RxPackets=0 RxBytes=8 RxFlagsSeen=0x00 LastRxReport=0 TxPackets=0 TxBytes=0 TxFlagsSeen=0x1e LastTxR
TCP IN 12.91.212.200:44194 -> 10.0.0.246:80 expires=17275421 RxPackets=115638 RxBytes=14280861 RxFlagsSeen=0x1e LastRxReport=17254328 TxPackets=118303 TxBytes=28101171 TxFlagsS
TCP OUT 12.91.212.202:30007 -> 12.91.212.200:42722 service expires=17274983 RxPackets=0 RxBytes=8 RxFlagsSeen=0x00 LastRxReport=0 TxPackets=0 TxBytes=0 TxFlagsSeen=0x1e LastTxR
```

Q&A