

NFF-GO (YANFF) — YET ANOTHER NETWORK FUNCTION FRAMEWORK LABS

Intel Golang Team

Gregory Shimansky

Legal Information

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

No computer system can be absolutely secure.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors.

Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit http://www.intel.com/performance.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit http://www.intel.com/performance.

Cost reduction scenarios described are intended as examples of how a given Intel- based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel, the Intel logo and others are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

© 2017 Intel Corporation.





YANFF - Yet Another Network Function Framework

Framework for building performant native network functions

- Open-source project
- Higher level abstractions than DPDK
- Go language: productivity, performance, concurrency, safety
- Network functions are application programs and not virtual machines

Benefits:

- Easily leverage IA HW capabilities: multi-cores, AES-NI, CAT, QAT, DPDK
- 10x reduction lines of code
- No need to be expert network system programmer
- Similar performance with C
- Take advantage of cloud native deployment: continuous delivery, microservices, containers

https://github.com/intel-go/yanff



Technical Motivation

- Developers need framework to shorten development cycle of VNFs
 - Currently VNFs are monolithic "virtual appliances" instead of network functions
 - Significant part of VNF is about plumbing. Plumbing VNFs to CommSPs network is an art. Should be abstracted from VNFs
- Lack of stable and unified APIs for VNF control and data plane
- Challenges with access to HW Accelerators in cloud environment.
- Cloud-friendly APIs and designs needed.

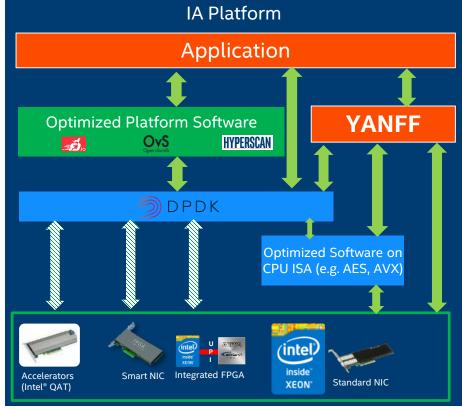
Accelerating transition to from rule-based networking to imperative networking



YANFF: Yet Another Network Function Framework

- Simple but powerful abstractions:
 - Flow, Packet
- User builds packet processing graph using "flow functions" chaining
 - SetReceiver -> SetHandler -> SetSender
 - Several predefined possibilities of adding user processing inside packet processing graph
 - Split, Separate, Generate, Handle
- Can leverage predefined functions which parse packets, check ACL rules, etc.
- Run to completion NFs can be expressed in the flow functions and natural chaining
- Auto-scaling, ease of development
- Zero-copy between NFs
- Flexible incoming flow handling sources can be anything: network port, memory buffer, remote procedure call, etc.







L3 Simple Forwarding Example

```
var L3Rules *rules_L3Rules
func main() {
     flow.SystemInit(16)
     L3Rules = rules.GetL3RulesFromORIG("Forwarding.conf")
     inputFlow := flow.SetReceiver(0)
     outputFlows := flow.SetSplitter(inputFlow, L3Splitter, uint(3))
     flow.SetStopper(outputFlows[0])
     for i := 1; i < 3; i++ {
          flow.SetSender(outputFlows[i], uint8(i-1))
     flow.SystemStart()
// User defined function for splitting packets
func L3Splitter(currentPacket *packet.Packet) uint {
     currentPacket.ParseL4()
     return rules.L3_ACL_port(currentPacket, L3Rules)
```



Configuration file for Forwarding

Source address, Destination address, L4 protocol ID, Source port, Destination port, Output port

111.2.0.0/31	ANY	tcp	ANY	ANY	1
111.2.0.2/32	ANY	tcp	ANY	ANY	Reject
ANY	ANY	udp	3078:3964	56:61020	2



Exactly The Same Example in DPDK/C

```
#include catdlib.ho
#include catdint.ho
                                                                                                                                                                                                                                                                       prepare_one_packet(struct_rte_mbuf **pkts_in, struct_acl_search_t *acl,
                                                                                                                                                                                          RTE ACL IPVAVLAN PROTO,
                                                                                                                                                                                                                                                                              int index)
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               /* Initialize TX buffers */
#include cays/types.ho
                                                                                                                                                                                          RTE ACL IPWIVLAN DST.
#include catdarg.h>
                                                                                                                                                                                          RTE ACL IPVMVLAN PORTS
                                                                                                                                                                                          RTE ACL IPWIVLAN NUM
                                                                                                                                                                                    truct rte acl field def ipv4 defx[NLM FIELDS IPV4] = {
                                                                                           static uint16 t nb lcore params = sizeof(lcore params array default) /
                                                                                                                                                                                                                                                                                       /* Check to make sure the packet is valid (RFC1B12) *
#include orte log.bo
                                                                                                                                                                                                  .type = RTE_ACL_FIELD_TYPE_BITMASK,
                                                                                            static struct rte eth conf port conf = {
                                                                                                                                                                                                   field index - PROTO FIFLD IPWA
#include orte memzone.ha
                                                                                                                                                                                                                                                                                               ex(tool hdr-shdr checksum):
                                                                                                            .split hdr size = 0,
 #include orte launch.ho
                                                                                                            .header_split = 0, /**c Header Split disabled */
.hw_ip_checksum = 1, /**c IP checksum offload enabled */
.hw_vlan_filter = 0, /**c VLAN filtering disabled */
                                                                                                                                                                                                                                                                                               acl->data ipv4[acl->num ipv4] = MBUF IPV4 2PROTO(pkt);
 #include orte atomic.ho
                                                                                                                                                                                                  .type - STE ACL FIELD TYPE MASK.
                                                                                                                                                                                                                                                                                               acl-om ipv4[(acl-onum ipv4)++] = pkt
#include orte prefetch.ho
                                                                                                                                                                                                   .field_index = SRC_FIELD_IPV4,
.input_index = RTE_ACL_IPV4VLAN_SRC.
                                                                                                             Jumbo frame . 0. /**c Jumbo Frame Support disabled */
                                                                                                             hw strip orc = 1, /**c CRC stripped by hardware */
                                                                                                                                                                                                    offset = offsetof(struct ipv4 hdr, src addr) -
offsetof(struct ipv4 hdr, nest proto id),
#include orte branch prediction.ht
                                                                                                   arx_adv_conf = 4
                                                                                                                                                                                                                                                                              } else if (RTI_ETH_IS_IPV6_HDR(pkt->packet_type)) {
 #include orte_pci.ho
                                                                                                                   .rss_key = MULL,
.rss_hf = ETH_RSS_EP | ETH_RSS_UDP |
                                                                                                                                                                                                  .type = RTE ACL FIELD TYPE MASK.
                                                                                                                                                                                                                                                                                       /* Fill acl structure */
#include orte random.ho
                                                                                                                                                                                                                                                                                       acl-odata inv6[acl-onum inv6] = MBUF IPV6 2PROTD(okt);
                                                                                                                           ETH RSS TCP | ETH RSS SCTP,
                                                                                                                                                                                                   .field_index = DST_FIELD_IPV4,
.input_index = RTE_ACL_IPV4VLAN_DST.
#include onto ethdey.ho
                                                                                                                                                                                                    offset = offsetof(struct ipv4_hdr, dst_addr) -
offsetof(struct ipv4_hdr, nest_proto_id),
#include crte_mempool.h:
#include crte_mbuf.h>
                                                                                                    .txmode = {
                                                                                                                                                                                                                                                                                       /* Unknown type, drop the packet */
                                                                                                            .ng mode - ETH MD TX NONE.
#include crte ip.h>
#include orte string fns.ha
                                                                                                                        23 SLOC in YANFF vs 2079 in DPDK/C!
 #1# RTC_LOG_LEVEL >= RTC_LOG_DEBUG
 #define LiFWDACL DEBUG
 #define DO_RFC_1812_CHECKS
 Adefine MAX JUMBO PAT LEN 9500
                                                                                           #define DEFAULT MAX CATEGORIES 1
                                                                                           #define LSTND ACL IPV4 NAME
                                                                                                                                                                                                                                                                              ) else if (RTE ETH IS IPVS HDR(pkt->packet type))
 #define MEMPOOL CACHE SIZE 256
                                                                                                                                                                                                          sizeof(wint16 t).
                                                                                                                                                                                                                                                                                       /* Fill acl structure */
acl->data ipv6[acl->num_ipv6] = MBUF_IPV6_2PROTO(pkt);
                                                                                          #define ROUTE LEAD CHAR
                                                                                                                                                                                                                                                                                      acl->m ipv6[(acl->num ipv6)++] = pkt;
    This expression is used to calculate the number of mbufs needed
                                                                                          Adation COMMENT LEAD CHAR
                                                                                                                                                                                #define IPV6 ADDR UI6 (IPV6 ADDR LEN / sizeof(uint16 t))
#define IPV6 ADDR U32 (IPV6 ADDR LEN / sizeof(uint32 t))
                                                                                          #define OPTION NONLMA
    RTE MAX is used to ensure that NO MOUF never goes below a
                                                                                          #define OPTION ENDING
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       fflush(stdout);
                                                                                                                                                                                                                                                                       fendif /* DO RFC 1812 CHECKS */
                                                                                          #define OPTION_SCALAR
                                                                                          #define ACL DENY SIGNATURE
          (nb_portx * nb_rx_queue*RTE_TEST_RX_DESC_DEFAULT +
                                                                                          Admittee BTC LOCTVDC LICHDACI
                                                                                                                           BTT LOCTURE USERS
                                                                                                                                                                                                                                                                       prepare acl parameter(struct rte mbuf ""pkts in, struct acl search t "acl.
                                                                                                                                                                                          SRC3 FIELD IPV6,
          nb ports * nb lcores * MAX PKT BURST +
                                                                                                                                                                                          SRC4 FIELD IPV6
          nb ports * n tx queue * RTE TEST TX DESC DEFAULT +
                                                                                            #define wint32 t to char(ip, a, b, c, d) do (\
                                                                                                            *a = (unsigned char)(ip >> 24 & @xff);
                                                                                                                                                                                          DST3 FIELD IPV6
                                                                                                                                                                                          DSTA FIELD TRV6
                                                                                                            "d = (unsigned char)(ip & 0xff);\
#define BURST_TX DRAIN US 100 /* TX drain every -100ux */
                                                                                                                                                                                          MUM FIELDS IPVS
                                                                                                                                                                                                                                                                              for (1 = 0: 1 c PETETON OFFSET SS 1 c ph re: 144) /
                                                                                                                                                                                                                                                                                       rte_prefetch@(rte_pktmbuf_mtod(
pkts_in[i], void *));
                                                                                            #define OFF_IPV62PROTO (offsetof(struct ipv6_hdr, proto))
/* Configure how many packets ahead to prefetch, when reading packets */
                                                                                                                                                                                     uct rte acl field def ipv6 defx[MLM_FIELDS_IPV6] = {
                                                                                                     rte_pktmbuf_mtod_offset((m), uint&_t *, OFF_ETHNEAD + OFF_IPV42FROTO
                                                                                                                                                                                                 type - STE ACL FIELD TYPE BITMASK
                                                                                                    rte oktobut stod offset((s), wint& t *, OFF ETHERAD + OFF IPV62PROTO
                                                                                                                                                                                                  .mize = mizeof(uintE t),
.field index = PROTO FIELD IPV6,
                                                                                                                                                                                                   input index - PROTO FIELD IPV6.
                                                                                                                                                                                                                                                                                      prepare one packet(pkts in, acl, i);
 #define RTC TEST RX DESC DEFAULT 128
                                                                                                   unsigned long val;
                                                                                                    char "end;
errno = 0;
 static wint16 t nb rxd = RTC TEST RX DESC DEFAULT;
                                                                                                                                                                                                  .type = RTE ACL FIELD TYPE MASK,
 static wint16 t nb txd = RTE TEST TX DESC DEFAULT:
                                                                                                   if (errso != 0 || end[0] != (dlm) || val > (lim))
                                                                                                                                                                                                                                                                                      prepare one packet(pkts in, acl, i);
                                                                                                   (fd) = (typeof(fd))val;
(in) = end + 1;
 static struct ether addr ports eth addr(RTE MAX ETHPORTS):
                                                                                                                                                                                                           offsetof(struct ipv6 hdr. proto
                                                                                                                                                                                                                                                                       send one packet(struct rte mbuf *m. uint32 t res)
static uint32 t enabled port mask;
1 static int promiscuous on; /**< Ports set in promiscuous mode off by defaul
2 static int nums on = 1; /**< NUMN is enabled by default. */</p>
                                                                                                                                                                                                  .type = RTE ACL FIELD TYPE MASK.
                                                                                              * always be found when input packets have multi-matches in the database.
                                                                                               A exception case is performance measure, which can define route rules wi
                                                                                                                                                                                                                                                                                               (wintS t)(res - FWD PORT SHIFT)):
                                                                                               higher priority and route rules will always be returned in each lookup. 
Reserve range from ACL RULE PRIORITY MAX + 1 to
                                                                                                                                                                                                   offset = offsetof(struct inv6 bdc, acc addc)
                                                                                                                                                                                                                                                                                      /* in the ACL list, drop it */
```

if (qconf->tx_buffer[portid] == NULL)
 rte_exit(EXT_FAILURE, "Can't allocate tx buffer for port %u\n", for (long id = 0: long id < RTE MAX LCORE: long id++) { if (rte_lcore_ix_enabled(lcore_id) == 0) socketid = (wint& t) rte lopre to socket id(lopre id): spcketid = 0: printf("txq=%u,%d,%d", lcore_id, queueid, socketid); rte_eth_dev_info_get(portid, &dev_info); txconf = Edev info,default txconf; if (port conf.rxmode.jumbo frame) ret = rte_eth_tx_queue_xetup(portid, queueid, nb_txd, "rte eth tx queue setup: errolid. gconf->tx queue id[portid] = queueid: gconf->tx port id[gconf->n tx port] = portid; if (rte lopre is enabled(lopre id) == 0) printf("\nInitializing rx queues on lcore %u ... ", lcore_id); portid = qconf->rx queue list[queue].port_id; consists a confuser come list(come) come idspcketid = (wintE t) ste long to socket (dillong (d)) socketid = 0; printf("rxq"%d,%d,%d ", portid, queueid, socketid); ret = rte_eth_rx_queue_setup(portid, queueid, nb_rxd, pktmbuf_pool(socketid)); rte exit(EXIT FAILURE, "port:%d\n", ret, portid); for (portid = 0; portid < nb ports; portid++) { if ((enabled port mask & (1 << portid)) == 0) /* Start device */ ret = rte eth dev start(portid); rts_exit(EXIT_FAILURE, "rte eth dev start; erro%d, porto%d\n".

RTE ETH TX BUFFER SIZE(MAX PXT BURST), 0.



YANFF – Main Architectural Concepts

Flow

Abstraction without public fields, which is used for pointing connections between Flow functions.

Opened by Receive / Split / Separate / Counter / Generate.

Closed by Send / Merge / Stop.

Port Network door, used in Receive, Send.

Packet

High-level representation of network packet. Private field is *mbuf, public fields are mac / ip / data /etc: pointers to mbuf with offsets (zero copy).

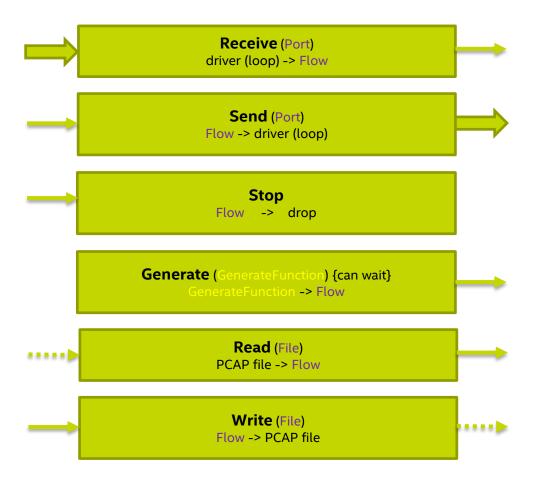
Is extracted before any User defined function. Can be filled after user request by Packet functions. Can be checked by Rule functions.

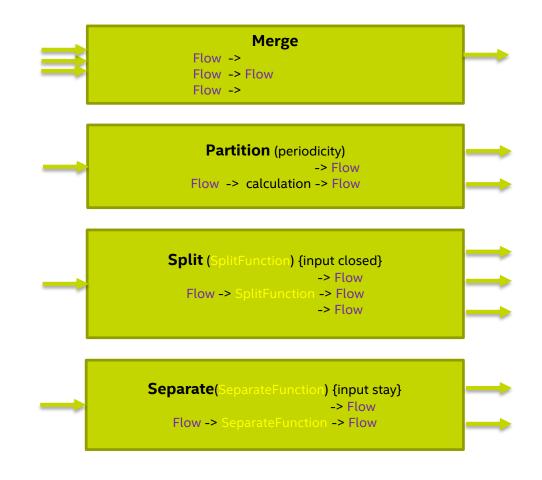
Rule

Set of checking rules, used in User defined functions.



Building Processing Graph







Packet modification functions

Handle (SeparateFunction) {can drop} -> Stop Flow -> SeparateFunction-> Flow Handle (HandleFunction) {can't drop} Flow -> HandleFunction -> Flow

Packet functions

Parsing packet fields

Parse L2 or/and L3 or/and L4 levels

Initializing packet fields

Initialize L2 or/and L3 or/and L4 levels

Encapsulate / Decapsulate

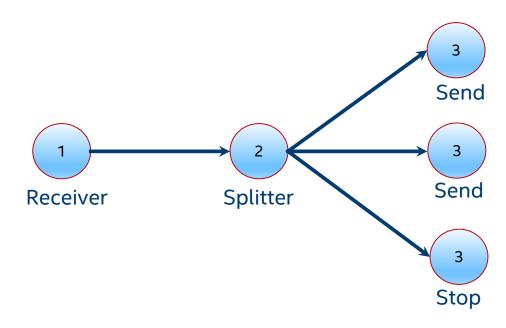
Rule functions

Create rule

Create checking rule from json / config

Checking packet fields by rule Check L2 or/and L3 or/and L4 levels

Flow Graph Example - Forwarding



Let's build some functions!

Create test VMs

- 1.Create and provision two test VMs:
 - \$ cd nff-go/vagrant
 - \$ vagrant up
- 2. Open two terminal windows
- 3.cd to vagrant directory below
- 4.run "vagrant ssh nff-go-"<a href="VM_number" to connect to pktgen VM and target VM, e.g.

```
$ vagrant ssh nff-go-1  # NFF-Go test program host
yanff-1$ bindports  # if ports not bound yet
```

```
$ vagrant ssh nff-go-0  # pktgen host
yanff-0$ bindports  # if ports not bound yet
```

Let's try (01 of 11)

```
Flow graph:
nff-go-1$ cd $YANFF/examples/tutorial }
nff-qo-1$ sudo ./step01
nff-go-0$ cd $YANFF/examples/tutorial
nff-qo-0$ ./qenscripts
nff-go-0$ ./runpktgen.sh
Pktgen:/> start 0
Pktgen:/> quit
```

```
// Init YANFF system
config := flow.Config{}
checkFatal(flow.SystemInit(&config))
initCommonState()
checkFatal(flow.SystemStart())
```

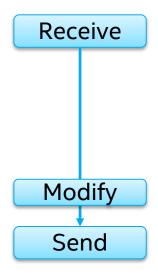
import "github.com/intel-go/yanff/flow"

package main

func main() {

Let's try (02 of 11)

Flow graph:



nff-go-1\$ sudo ./step02

```
nff-go-0$ ./runpktgen.sh
Pktgen:/> load step02.pg
Pktgen:/> start 0
...
Pktgen:/> quit
```

```
package main
import "github.com/intel-go/yanff/flow"
func main() {
    config := flow.Config{}
    checkFatal(flow.SystemInit(&config))
    initCommonState()
    firstFlow, err := flow.SetReceiver(0)
    checkFatal(err)
    checkFatal(flow.SetHandler(firstFlow, modifyPacket[0], nil))
    checkFatal(flow.SetSender(firstFlow, 0))
    checkFatal(flow.SystemStart())
```

Let's try (03 of 11)

Flow graph:

```
Partition

Modify

Send

Send
```

```
nff-go-1$ sudo ./step03
```

```
nff-go-0$ ./runpktgen.sh
Pktgen:/> load step03.pg
Pktgen:/> start 0
```

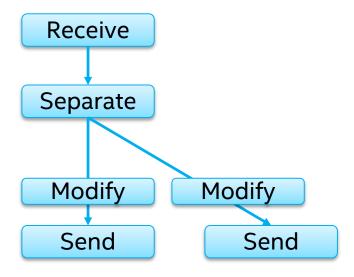
•••

Pktgen:/> quit

```
package main
import "github.com/intel-go/yanff/flow"
func main() {
    config := flow.Config{}
    checkFatal(flow.SystemInit(&config))
    initCommonState()
    firstFlow, err := flow.SetReceiver(0)
    checkFatal(err)
    secondFlow, err := flow.SetPartitioner(firstFlow, 300, 300)
    checkFatal(err)
    checkFatal(flow.SetHandler(firstFlow, modifyPacket[0], nil))
    checkFatal(flow.SetHandler(secondFlow, modifyPacket[1], nil))
    checkFatal(flow.SetSender(firstFlow, 0))
    checkFatal(flow.SetSender(secondFlow, 1))
    checkFatal(flow.SystemStart())
```

Let's try (04 of 11)

Flow graph:



```
nff-qo-1$ sudo ./step04
```

```
nff-go-0$ ./runpktgen.sh
Pktgen:/> load step04.pg
Pktgen:/> start 0
...
Pktgen:/> quit
```

```
package main
import "github.com/intel-go/yanff/flow"
import "github.com/intel-go/yanff/packet"
func main() {
     config := flow.Config{}
     checkFatal(flow.SystemInit(&config))
     initCommonState()
    firstFlow, err := flow.SetReceiver(0)
     checkFatal(err)
     secondFlow, err := flow.SetSeparator(firstFlow, mySeparator, nil)
     checkFatal(err)
     checkFatal(flow.SetHandler(firstFlow, modifyPacket[0], nil))
     checkFatal(flow.SetHandler(secondFlow, modifyPacket[1], nil))
     checkFatal(flow.SetSender(firstFlow, 0))
     checkFatal(flow.SetSender(secondFlow, 1))
     checkFatal(flow.SystemStart())
func mySeparator(cur *packet.Packet, ctx flow.UserContext) bool {
    cur.ParseL3()
    if cur.GetIPv4() != nil {
         cur.ParseL4ForIPv4()
         if cur.GetTCPForIPv4() != nil &&
packet.SwapBytesUint16(cur.GetTCPForIPv4().DstPort) == 53 {
               return false
    return true
```

Let's try (05 of 11) "" "" "github.com/intel-go/yanff/rules"

Flow graph:

```
Receive Rules

Separate Modify

Send Send
```

```
nff-go-1$ sudo ./step05
```

```
nff-go-0$ ./runpktgen.sh
Pktgen:/> load step05.pg
Pktgen:/> start 0
```

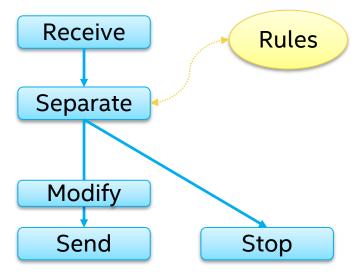
•••

```
Pktgen:/> quit
```

```
var L3Rules *rules.L3Rules
func main() {
    var err error
    config := flow.Config{}
    checkFatal(flow.SystemInit(&config))
    initCommonState()
    l3Rules, err = packet.GetL3ACLFromORIG("rules1.conf")
    checkFatal(err)
    firstFlow, err := flow.SetReceiver(0)
    checkFatal(err)
    secondFlow, err := flow.SetSeparator(firstFlow, mySeparator, nil)
    checkFatal(err)
    checkFatal(flow.SetHandler(firstFlow, modifyPacket[0], nil))
    checkFatal(flow.SetHandler(secondFlow, modifyPacket[1], nil))
    checkFatal(flow.SetSender(firstFlow, 0))
    checkFatal(flow.SetSender(secondFlow, 1))
    checkFatal(flow.SystemStart())
}
func MySeparator(cur *packet.Packet, ctx flow.UserContext) bool {
    return cur.L3ACLPermit(l3Rules)
```

Let's try (06 of 11) "......" func main() {

Flow graph:



```
nff-go-1$ sudo ./step06
```

```
nff-go-0$ ./runpktgen.sh
Pktgen:/> load step06.pg
Pktgen:/> start 0
```

•••

Pktgen:/> quit

```
var err error
    config := flow.Config{}
    checkFatal(flow.SystemInit(&config))
    L3Rules = rules.GetL3RulesFromORIG("rules1.conf")
    checkFatal(err)
    firstFlow, err := flow.SetReceiver(0)
    checkFatal(err)
    secondFlow, err := flow.SetSeparator(firstFlow, mySeparator, nil)
    checkFatal(err)
    checkFatal(flow.SetHandler(firstFlow, modifyPacket[0], nil))
    checkFatal(flow.SetSender(firstFlow, 0))
    checkFatal(flow.SetStopper(secondFlow))
    checkFatal(flow.SystemStart())
func MySeparator(cur *packet.Packet, ctx flow.UserContext) bool {
    return cur.L3ACLPermit(l3Rules)
```

Let's try (07 of 11)

```
Flow graph:
                    updated
   Receive
                     Rules
  Separate
   Modify
                  Stop
    Send
nff-go-1$ sudo ./step07
nff-go-0$ ./runpktgen.sh
Pktgen:/> load step07.pg
Pktgen:/> start 0
```

```
import "time"
var rulesp unsafe.Pointer
    l3Rules, err := packet.GetL3ACLFromORIG("rules1.conf")
    checkFatal(err)
    rulesp = unsafe.Pointer(&l3Rules)
    go updateSeparateRules()
... ... ...
func MySeparator(cur *packet.Packet, ctx flow.UserContext) bool {
    localL3Rules := (*packet.L3Rules)(atomic.LoadPointer(&rulesp))
    return cur.L3ACLPermit(localL3Rules)
func updateSeparateRules() {
    for {
        time.Sleep(time.Second * 5)
        locall3Rules, err := packet.GetL3ACLFromORIG("rules1.conf")
        checkFatal(err)
        atomic.StorePointer(&rulesp, unsafe.Pointer(locall3Rules))
```

To make changes in rules1.conf file it is necessary to connect to target VM in another window or run YANFF executable in screen terminal multiplexer.



Pktgen:/> quit

Let's try (08 of 11)

Flow graph: updated Receive Rules Split Modify Modify Send Send Stop nff-qo-1\$ sudo ./step08 nff-go-0\$./runpktgen.sh Pktgen:/> load step08.pg Pktgen:/> start 0 Pktgen:/> quit

```
const flowN = 3
     firstFlow, err := flow.SetReceiver(0)
     checkFatal(err)
     outputFlows, err := flow.SetSplitter(firstFlow, mySplitter, flowN, nil)
     checkFatal(err)
     checkFatal(flow.SetStopper(outputFlows[0]))
     for i := uint8(1); i < flowN; i++ {
          checkFatal(flow.SetHandler(outputFlows[i], modifyPacket[i-1], nil))
          checkFatal(flow.SetSender(outputFlows[i], i-1))
func mySplitter(cur *packet.Packet, ctx flow.UserContext) uint {
     localL3Rules := L3Rules
     return cur.L3ACLPort(localL3Rules)
```

To make changes in rules2.conf file it is necessary to connect to target VM in another window or run YANFF executable in screen terminal multiplexer.



Let's try (09 of 11)

```
Flow graph:
                    updated
  Receive
                     Rules
    Split
      Handle
                  Modify
       Modify
 Stop
           Send
                      Send
nff-qo-1$ sudo ./step09
nff-go-0$ ./runpktgen.sh
Pktgen:/> load step09.pg
Pktgen:/> start 0
```

```
import "github.com/intel-go/yanff/common"
    firstFlow, err := flow.SetReceiver(0)
    checkFatal(err)
     outputFlows, err := flow.SetSplitter(firstFlow, mySplitter, flowN, nil)
    checkFatal(err)
    checkFatal(flow.SetStopper(outputFlows[0]))
    checkFatal(flow.SetHandler(outputFlows[1], myHandler, nil))
    for i := uint8(1); i < flowN; i++ {
          checkFatal(flow.SetHandler(outputFlows[i], modifyPacket[i-1], nil))
          checkFatal(flow.SetSender(outputFlows[i], i-1))
func myHandler(cur *packet.Packet, ctx flow.UserContext) {
    cur.EncapsulateHead(common.EtherLen, common.IPv4MinLen)
    cur.ParseL3()
    cur.GetIPv4NoCheck().SrcAddr = packet.BytesToIPv4(111, 22, 3, 0)
     cur.GetIPv4NoCheck().DstAddr = packet.BytesToIPv4(3, 22, 111, 0)
    cur.GetIPv4NoCheck().VersionIhl = 0x45
    cur.GetIPv4NoCheck().NextProtoID = 0x04
```

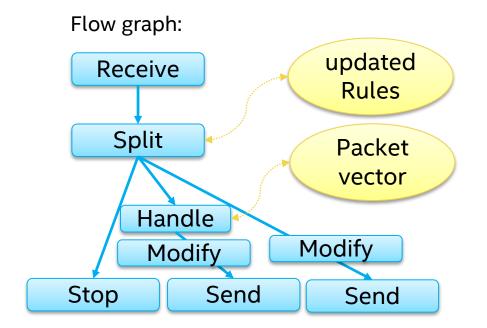
To make changes in rules2.conf file it is necessary to connect to target VM in another window or run YANFF executable in screen terminal multiplexer.



Pktgen:/> quit

Let's try (10 of 11)

...



```
nff-go-1$ sudo ./step10
```

```
nff-go-0$ ./runpktgen.sh
Pktgen:/> load step10.pg
Pktgen:/> start 0
...
```

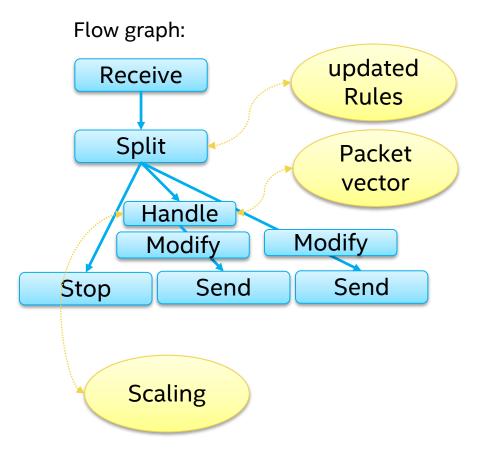
Pktgen:/> quit

```
func myHandler(curV []*packet.Packet, num uint, ctx flow.UserContext) {
    for i := uint(0); i < num; i++ {
        cur := curV[i]
        cur.EncapsulateHead(common.EtherLen, common.IPv4MinLen)
        cur.ParseL3()
        cur.GetIPv4NoCheck().SrcAddr = packet.BytesToIPv4(111, 22, 3, 0)
        cur.GetIPv4NoCheck().DstAddr = packet.BytesToIPv4(3, 22, 111, 0)
        cur.GetIPv4NoCheck().VersionIhl = 0x45
        cur.GetIPv4NoCheck().NextProtoID = 0x04
}</pre>
```

To make changes in rules2.conf file it is necessary to connect to target VM in another window or run YANFF executable in screen terminal multiplexer.



Let's try (11 of 11)



To make changes in rules2.conf file it is necessary to connect to target VM in another window or run YANFF executable in screen terminal multiplexer.

```
func myHandler(curV []*packet.Packet, num uint, ctx flow.UserContext) {
    for i := uint(0); i < num; i++ {
        cur := curV[i]
        cur.EncapsulateHead(common.EtherLen, common.IPv4MinLen)
        cur.ParseL3()
        cur.GetIPv4NoCheck().SrcAddr = packet.BytesToIPv4(111, 22, 3, 0)
        cur.GetIPv4NoCheck().DstAddr = packet.BytesToIPv4(3, 22, 111, 0)
        cur.GetIPv4NoCheck().VersionIhl = 0x45
        cur.GetIPv4NoCheck().NextProtoID = 0x04
    }
    // Some heavy computational code
    heavyCode()
}</pre>
```

```
nff-go-1$ sudo ./step11

nff-go-0$ ./runpktgen.sh
Pktgen:/> load step11.pg
Pktgen:/> start 0
...
Pktgen:/> quit
```



Alternative network packet IO

- KNI interfaces (examples/kni.go)
- Linux raw sockets (examples/OSforwarding.go)
- PCAP files (examples/clonablePcapDumper.go)
- Linux XDP (coming soon)

Statistic counters



```
// Set up address for stats web server
statsServerAddres = &net.TCPAddr{
        Port: 8080,
}

config := flow.Config{
        StatsHTTPAddress: statsServerAddres,
}
```

...



Finally: NAT

```
nff-go-1$ ./genscripts -pktgen direct
nff-go-1$ sudo ../nat/main/nat -config nat.json

nff-go-0$ ./runpktgen.sh
Pktgen:/> load nat.pg
Pktgen:/> start 0
Pktgen:/> start 1
...
Pktgen:/> quit
```

Q & A?

Optimization Notice

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2®, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804



Flow functions Receive (Port) driver (loop) -> Flow Send (Port) Flow -> driver (loop) Stop Flow -> free Merge {slow} Flow -> Flow -> Flow Partition (periodicity) Flow -> calculation -> Flow Split () {input closed} -> Flow Flow -> -> Flow -> Flow Separate() {input stay} -> Flow -> Flow Flow -> Handle () {can drop} -> Stop -> Flow Flow -> Handle () {can't drop} -> Flow Flow -> Generate () {can wait} -> Flow

Basic components

Instances (new types)

Flow

Abstraction without public fields, which is used for pointing connections between Flow functions.

Opened by Receive / Split /
Separate / Counter / Generate.
Closed by Send / Merge / Stop.

Packet

High-level representation of network packet. Private field is *mbuf, public fields are mac / ip / data /etc: pointers to mbuf with offsets (zero copy). Is extracted before any User defined function. Can be filled after user request by Packet functions. Can be checked by Rule functions.

Port

Network door, used in Receive, Send.

Rule

Set of checking rules, used in User defined functions.

Packet functions

Parsing packet fields Parse L2 or/and L3 or/and L4 levels

Initializing packet fields

Initialize L2 or/and L3 or/and L4 levels

Encapsulate / Decapsulate

Rule functions

Create rule

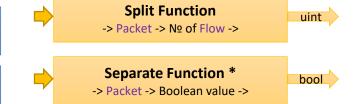
Create checking rule from json / config

Checking packet fields by rule Check L2 or/and L3 or/and L4 levels

Connections

- External (bytes inside network)
- Flow (*mbufs inside rings)
- Packets (as function arguments)

User defined functions



Handle Function *
-> Packet ->

Generate Function *
Packet ->

All functions take packet and handling context * Can process vector of packets at one time

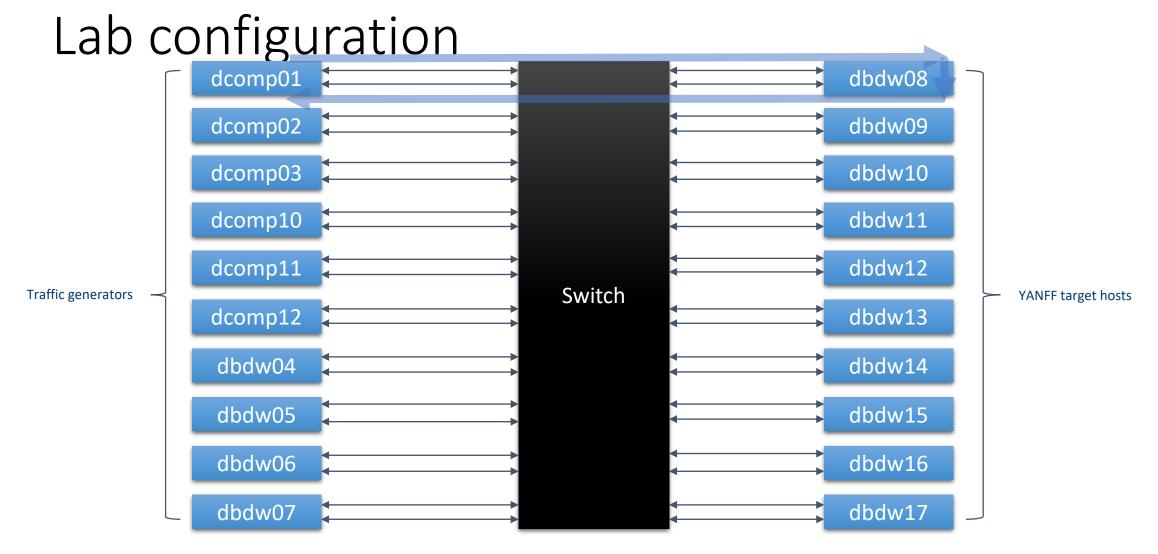
All functions at separate cores and can be cloned

Library External Components

- Flow: type "Flow" Init, Starting, Checking, Flow functions
- Packet: type "Packet", parsing / initializing packet functions
- Rules: type "Rule", parsing rules / checking Packet functions
- User package: user defined function.

Library Internal Components

- Scheduler: Cloning of user defined flow functions
- Asm: assembler functions added to GO
- Common: technical functions shared by other components
- Low: connections with DPDK C implementation



Jump host: , Login: gashiman, Password:

Finally (2 of 2): ipsec

Showing ipsec example