Library Operating System with Mainline Linux Network Stack

Hajime Tazaki, Ryo Nakamura, Yuji Sekiya

netdev0.1, Feb. 2015

Motivation

- Why kernel space ?
  - Packets were expensive in 1970’
- Why not userspace ?
  - well grown in decades, costs degrades
  - obtain network stack <u>personalization</u>
  - <u>controllable</u> by userspace utilities

Userspace network stacks

- A lot of userspace network stack
  - full scratch: mTCP, Mirage, lwIP
  - Porting: OSv, Sandstorm, libuinet (FreeBSD), Arrakis (lwIP), OpenOnload (lwIP?)
- Motivated by their own problems (specialized NIC, cloud, high-speed Apps)
- Writing a network stack is 1-week DIY,
  - but writing opera-table network stack is decades DIY (which is not DIY)

Questions

- How to benefit matured network stack in userspace ?
- How to trivially <u>introduce your idea</u> on network stack ?
  - xxTCP, IPvX, etc..
- How to flexibly <u>test your code</u> with a complex scenario ?

The answers

- Using Linux network stack <u>as-is</u>
- as <u>a userspace Library</u> (library operating system)

This talk is about

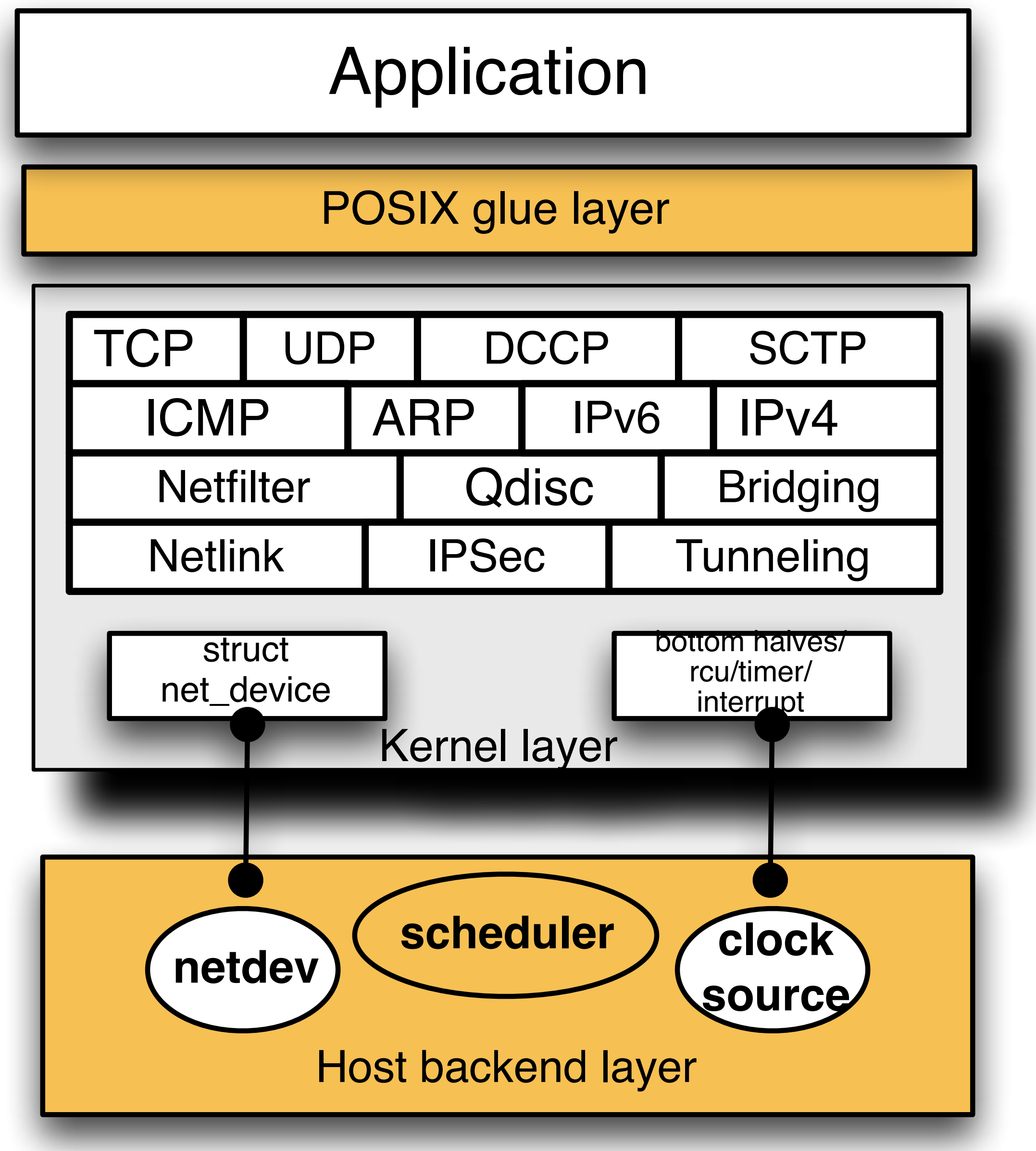
- an introduction of a library operating system for Linux
- and its implementation
- with a couple of useful use cases

Outlook (design)

- hardware-independent arch (arch/lib)
- 3 components
  - Host backend layer
  - Kernel layer
  - POSIX layer

https://github.com/libos-nuse/net-next-nuse

Outlook (cont'd)



4) applications magically runs

3) add POSIX glue code

1) Build Linux srctree w/ glues as a library

2) put backend (vNIC, clock source, scheduler) and bind

https://github.com/libos-nuse/net-next-nuse/blob/nuse/arch/lib/sched.c

Kernel glue code

```
void schedule(void)
{
    lib_task_wait();
}
signed long schedule_timeout(signed long timeout)
{
    u64 ns;
    struct SimTask *self;

    if (timeout == MAX_SCHEDULE_TIMEOUT) {
        lib_task_wait();
        return MAX_SCHEDULE_TIMEOUT;
    }
    lib_assert(timeout >= 0);
    ns = ((__u64)timeout) * (1000000000 / HZ);
    self = lib_task_current();
    lib_event_schedule_ns(ns, &trampoline, self);
    lib_task_wait();
    /* we know that we are always perfectly on time. */
    return 0;
}
```

https://github.com/libos-nuse/net-next-nuse/blob/nuse/arch/lib/nuse-glue.c

POSIX glue code

```
int nuse_socket(int domain, int type, int protocol)
{
    lib_update_jiffies();
    struct socket *kernel_socket = malloc(sizeof(struct socket));
    int ret, real_fd;

    memset(kernel_socket, 0, sizeof(struct socket));
    ret = lib_sock_socket(domain, type, protocol, &kernel_socket);
    if (ret < 0)
        errno = -ret;
 (snip)
    lib_softirq_wakeup();
    return real_fd;
}
weak_alias(nuse_socket, socket);
```

Implementations (Instances)

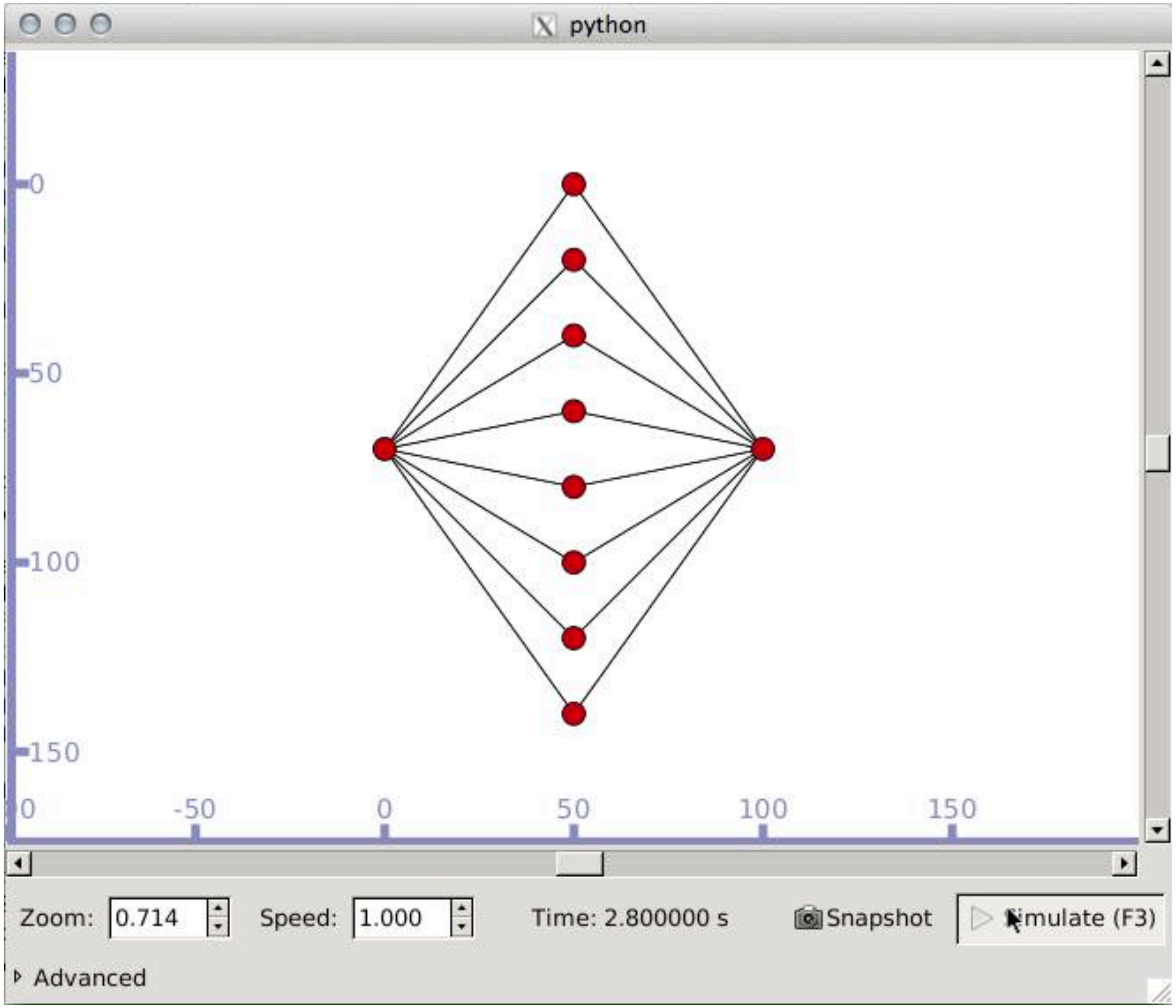
- Direct Code Execution (DCE)
  - network simulator integration (ns-3)
  - for more testing
- Network Stack in Userspace (NUSE)
  - gives new platform of Linux network stack
  - for ad-hoc network stack

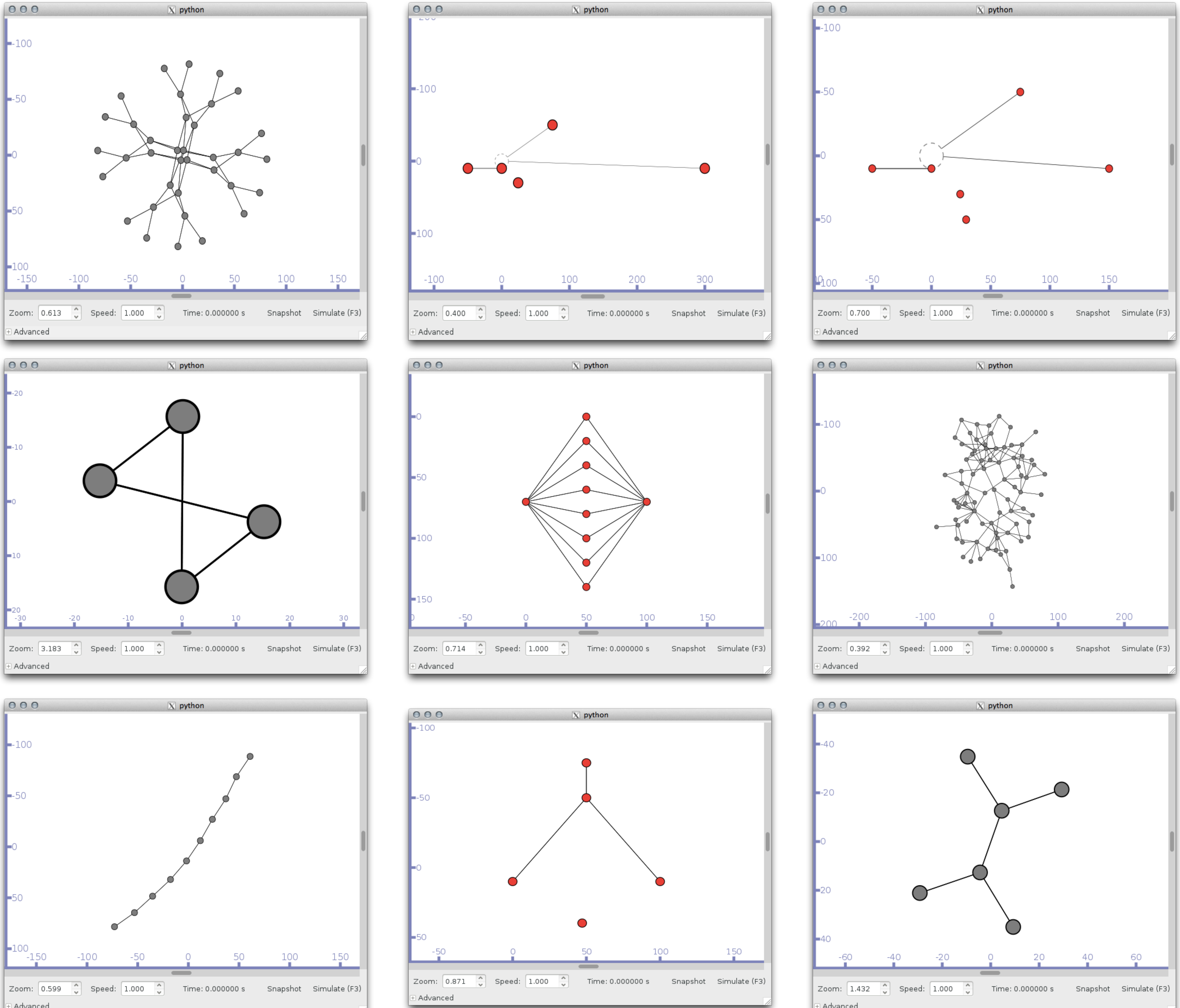
Direct Code Execution

- ns-3 integration
  - deterministic scheduler
- single-process model virtualization
  - dlmopen(3)-like virtualization
  - full control over multiple network stacks

Execution (DCE)

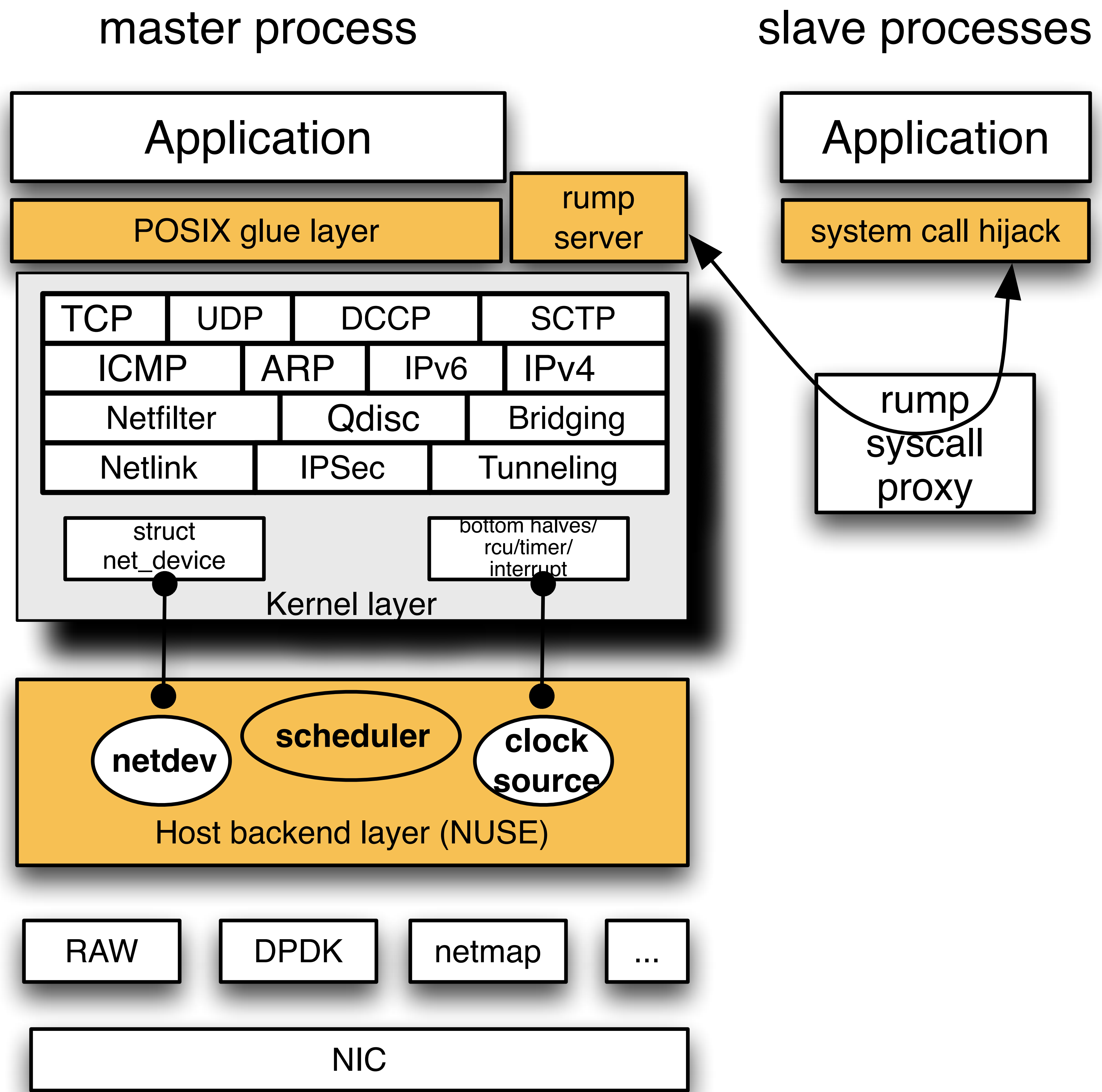
- main() => dlmopen(ping,liblinux.so)
  => main()=>socket(2)=>dce_socket()
  => (do whatever)


python
0
50
100
150
-50
0
50
100
150
Zoom: 0.714
Speed: 1.000
Time: 2.800000 s
Snapshot
Simulate (F3)
Advanced


python
Zoom: 0.613 Speed: 1.000 Time: 0.000000 s Snapshot Simulate (F3)
Advanced
python
Zoom: 0.400 Speed: 1.000 Time: 0.000000 s Snapshot Simulate (F3)
Advanced
python
Zoom: 0.700 Speed: 1.000 Time: 0.000000 s Snapshot Simulate (F3)
Advanced
python
Zoom: 3.183 Speed: 1.000 Time: 0.000000 s Snapshot Simulate (F3)
Advanced
python
Zoom: 0.714 Speed: 1.000 Time: 0.000000 s Snapshot Simulate (F3)
Advanced
python
Zoom: 0.392 Speed: 1.000 Time: 0.000000 s Snapshot Simulate (F3)
Advanced
python
Zoom: 0.599 Speed: 1.000 Time: 0.000000 s Snapshot Simulate (F3)
Advanced
python
Zoom: 0.871 Speed: 1.000 Time: 0.000000 s Snapshot Simulate (F3)
Advanced
python
Zoom: 1.432 Speed: 1.000 Time: 0.000000 s Snapshot Simulate (F3)
Advanced

Network Stack in Userspace

- Userspace network stack running on Linux (POSIX) platform
- Network stack personalization
- Full features by design (full stack)
    - ARP/ND, UDP/TCP (all cc algorithm), SCTP, DCCP, QDISC, XFRM, netfilter, etc.


master process
slave processes
Application
POSIX glue layer
rump server
Application
system call hijack
TCP
UDP
DCCP
SCTP
ICMP
ARP
IPv6
IPv4
Netfilter
Qdisc
Bridging
Netlink
IPSec
Tunneling
struct net_device
bottom halves/ rcu/timer/ interrupt
Kernel layer
rump syscall proxy
netdev
scheduler
clock source
Host backend layer (NUSE)
RAW
DPDK
netmap
...
NIC

Execution (NUSE)

- LD_PRELOAD=libnuse-linux.so \
  ping www.google.com
- ping(8) => socket(2) => nuse_socket() => raw(7) => (network)

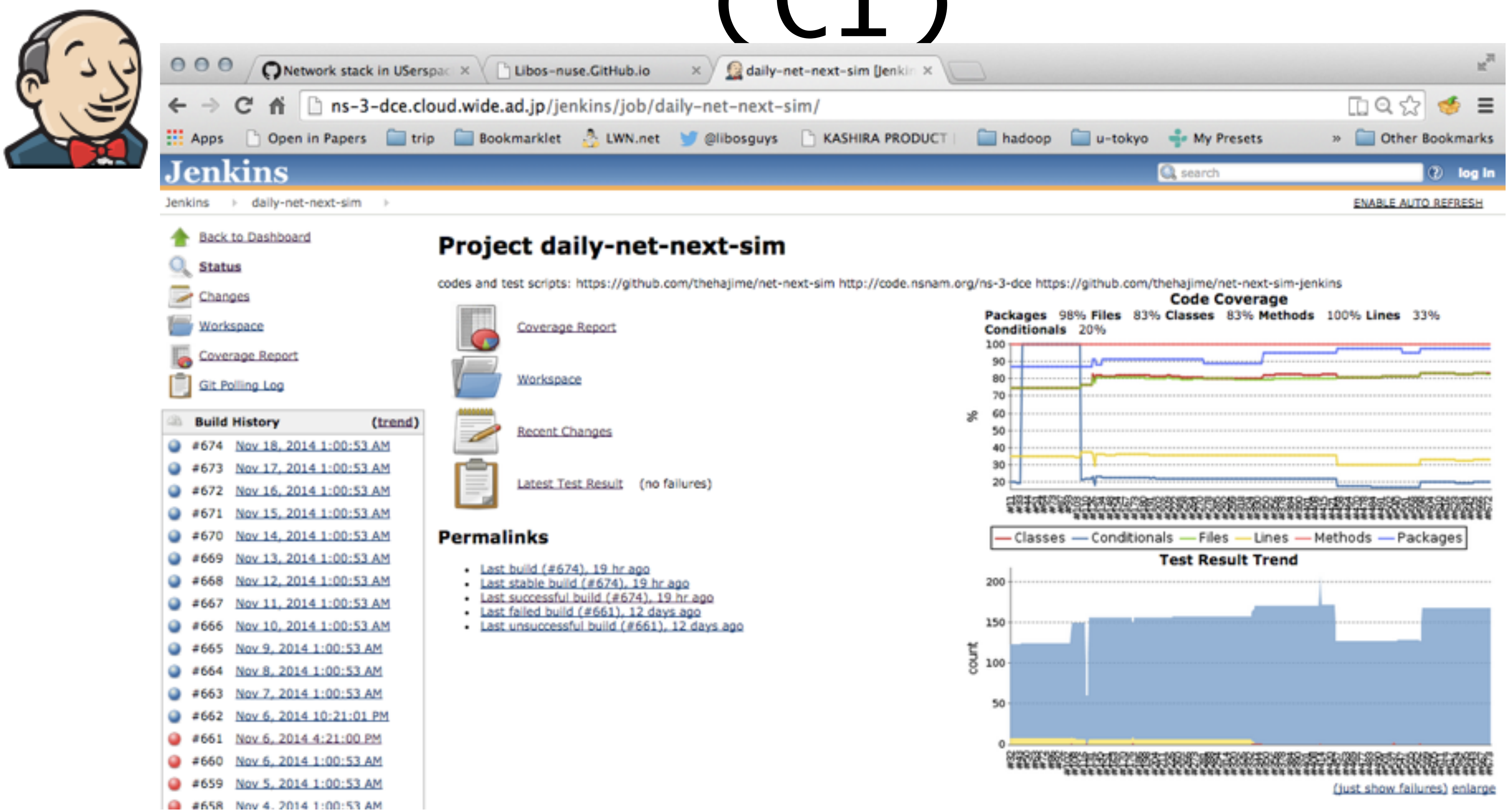
When it's useful?

- ad-hoc network stack (network stack personalization)
  - LD_PRELOAD=liblinux-mptcp.so firefox
- Bundle with kernel bypasses
  - Intel DPDK / netmap / PF_RING / etc.
- debugging/testing with ns-3

Testing workflow

1. Write/modify code (patches)
2. Write a test code (incl. packet exchanges)
3. if PASS; <u>accept pull-request</u> else; rejects

continuous integration (CI)



http://ns-3-dce.cloud.wide.ad.jp/jenkins/job/daily-net-next-sim/

T1) write a patch

```
Fixes: de3b7a06dfe1 ("xfrm6: Fix transport header offset in _decode_session6.")
Signed-off-by: Hajime Tazaki <tazaki@sfc.wide.ad.jp>
---
 net/ipv6/xfrm6_policy.c | 1 +
 1 file changed, 1 insertion(+)

diff --git a/net/ipv6/xfrm6_policy.c b/net/ipv6/xfrm6_policy.c
index 48bf5a0..8d2d01b 100644
--- a/net/ipv6/xfrm6_policy.c
+++ b/net/ipv6/xfrm6_policy.c
@@ -200,6 +200,7 @@ _decode_session6(struct sk_buff *skb, struct flowi *fl, int
reverse)

 #if IS_ENABLED(CONFIG_IPV6_MIP6)
        case IPPROTO_MH:
+           offset += ipv6_optlen(exthdr);
            if (!onlyproto && pskb_may_pull(skb, nh + offset + 3 - skb->data)) {
                struct ip6_mh *mh;
```

http://patchwork.ozlabs.org/patch/436351/

T2) write a test

```
                +-----------+
                |    HA     |
                +-----------+
                     |sim0
          +----------+------------+
          |sim0                   |sim0
   sim2+----+---+          +----+---+
    - -|  AR1   |          |  AR2   |
       +---+----+          +----+---+
           |sim1                |sim1
           |                    |

            sim0                  sim0
   +----+------+  (Movement) +----+-----+
   |    MR     |   <=====>   |    MR    |
   +-----------+             +----------+
       |sim1                     |sim1
   +---------+               +---------+
   |   MNN   |               |   MNN   |
   +---------+               +---------+
```

- As ns-3 scenario
  - C++ or python
  - create a topology
  - config nodes
  - run/check results (e.g., ping6)

http://code.nsnam.org/thehajime/ns-3-dce-umip/file/tip/test/dce-umip-test.cc

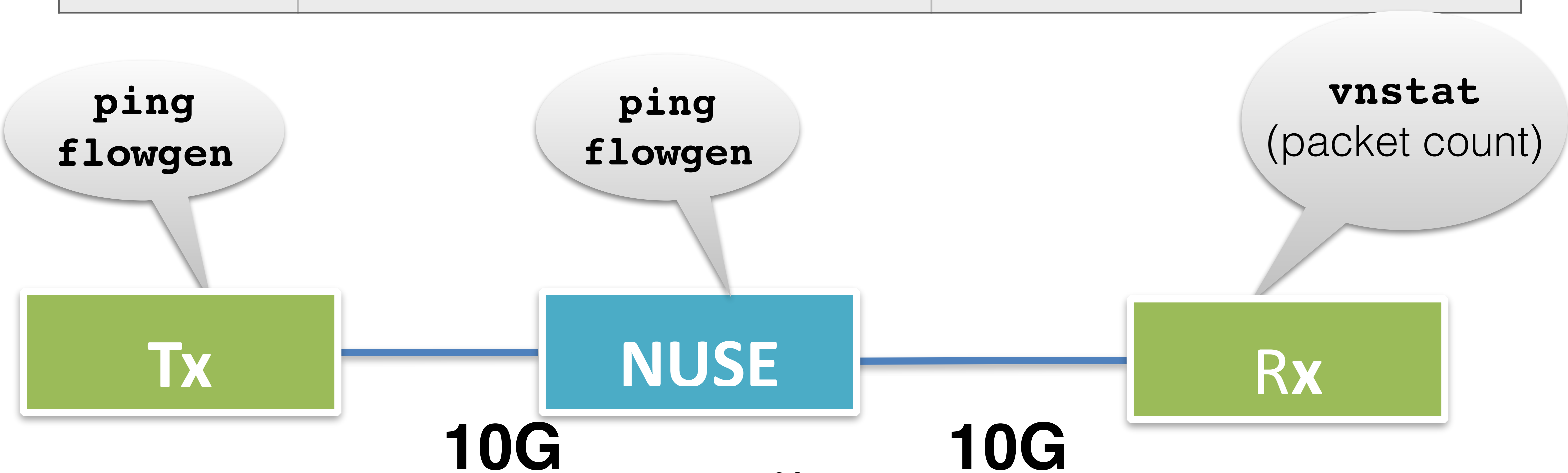
```
#!/usr/bin/python

from ns.dce import *
from ns.core import *

nodes = NodeContainer()
nodes.Create (100)
dce = DceManagerHelper()
dce.SetNetworkStack ("liblinux.so")
dce.Install (nodes)

app = DceApplicationHelper()
app.SetBinary ("ping6")
app.Install (nodes)
(snip)

NS_TEST_ASSERT_MSG_EQ (m_pingStatus, true, "Umip test " << m_testname
                       << " did not return successfully: " << g_testError)

Simulator.Stop (Seconds(1000.0))
Simulator.Run ()
```

Performance of NUSE

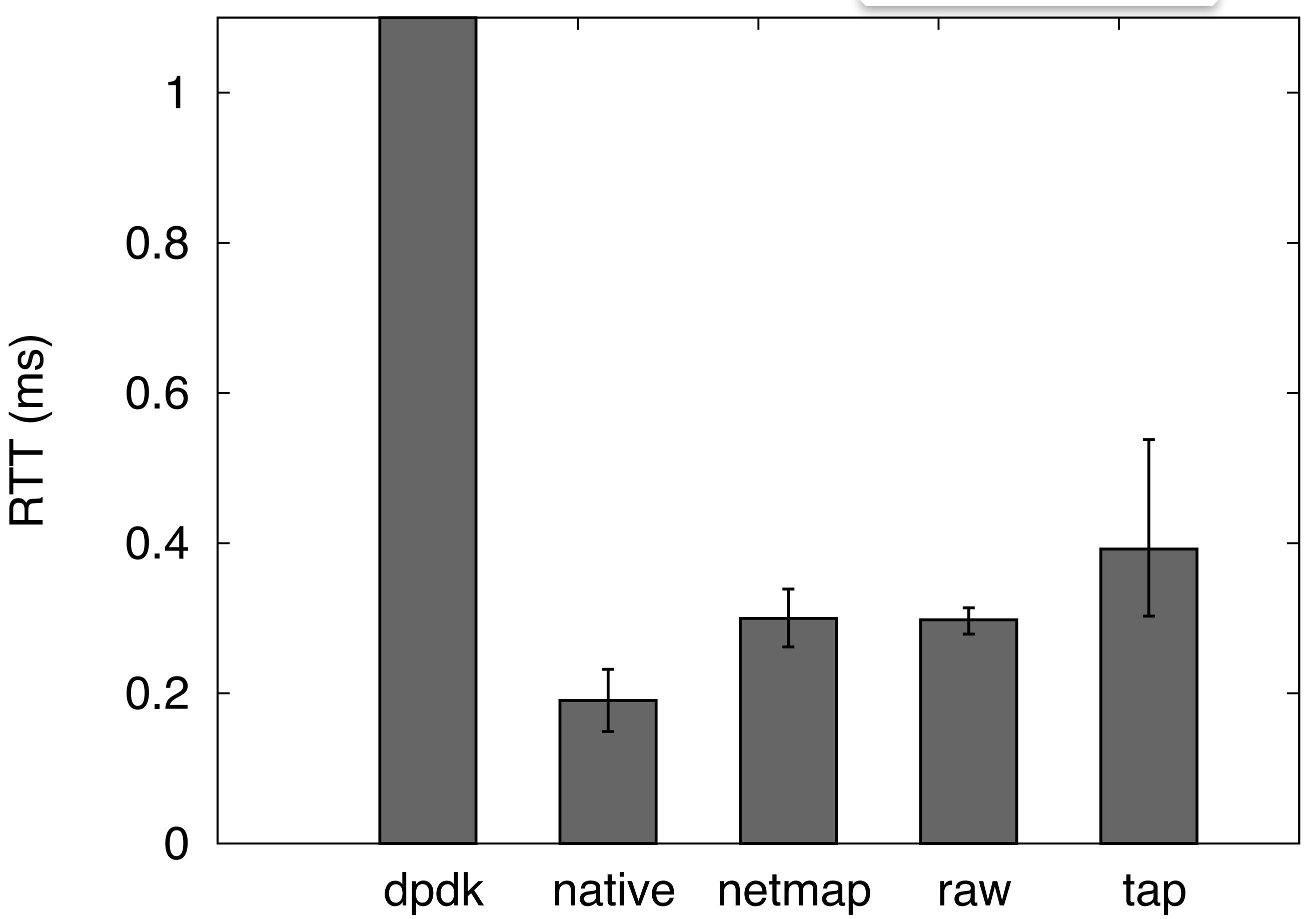
- 10G Ethernet back-to-back
  - transmission
  - IP forwarding
- native Linux, raw socket, tap, dpdk, netmap

Performance: setup

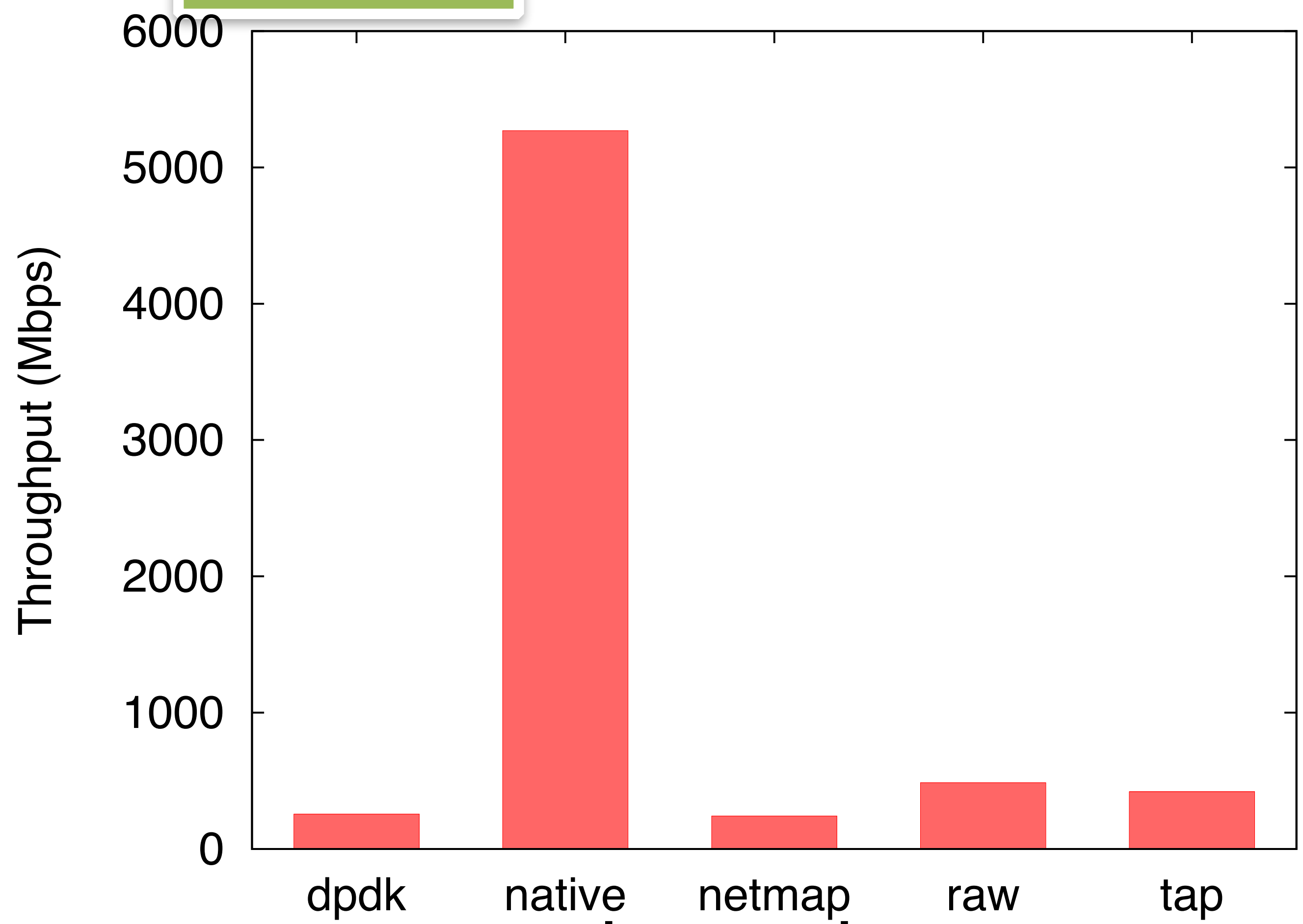
	NUSE node	Tx/Rx nodes
CPU	Xeon E5-2650v2 @ 2.60GHz (16 core)	Xeon L3426 @ 1.87GHz (8 core)
Memory	32GB	4GB
NIC	Intel X520	Intel X520
OS	host:3.13.0-32 nuse: 3.17.0-rc1	host:3.13.0-32



Host Tx



ping (RTT)

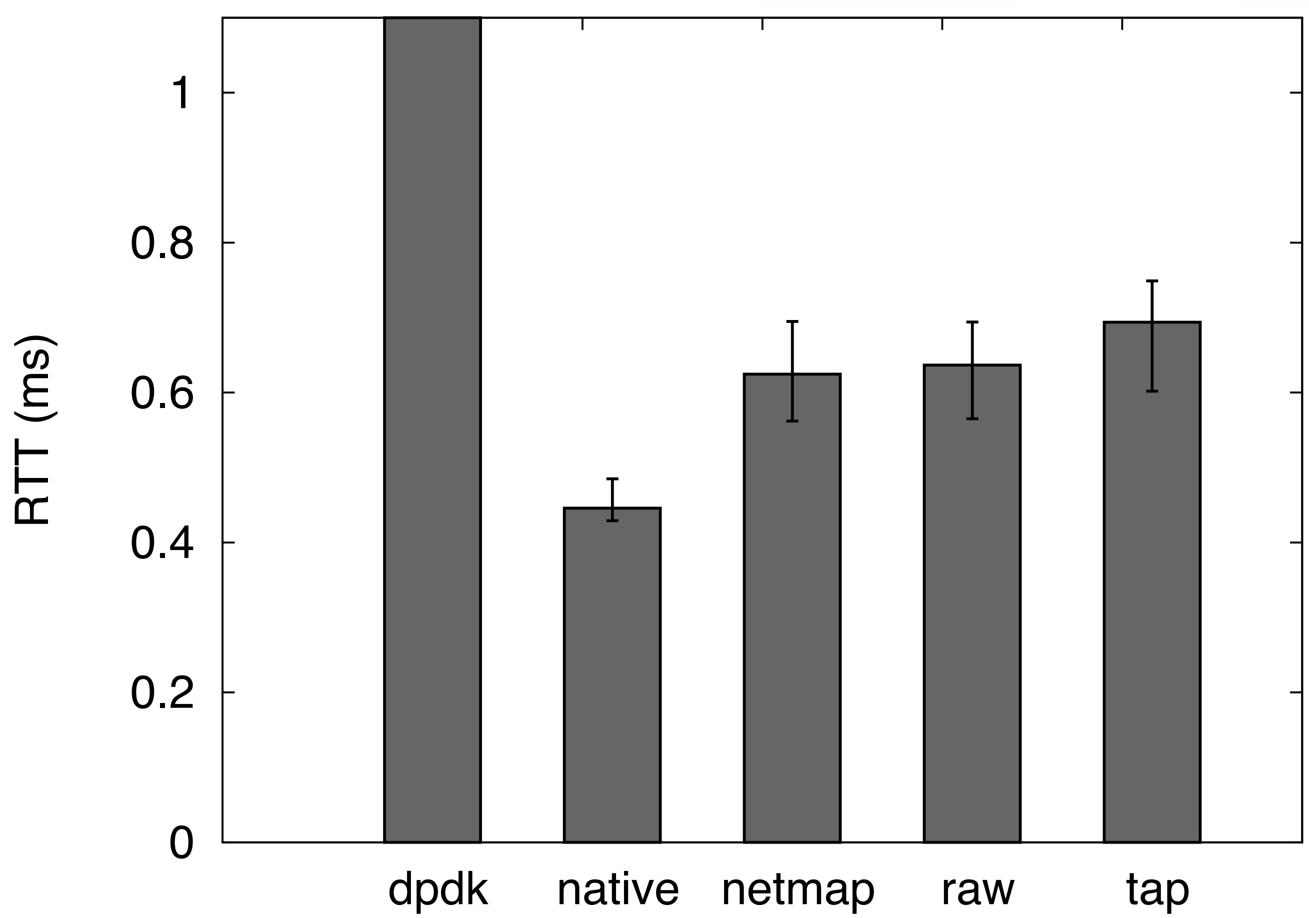


throughput
(1024byte,UDP)

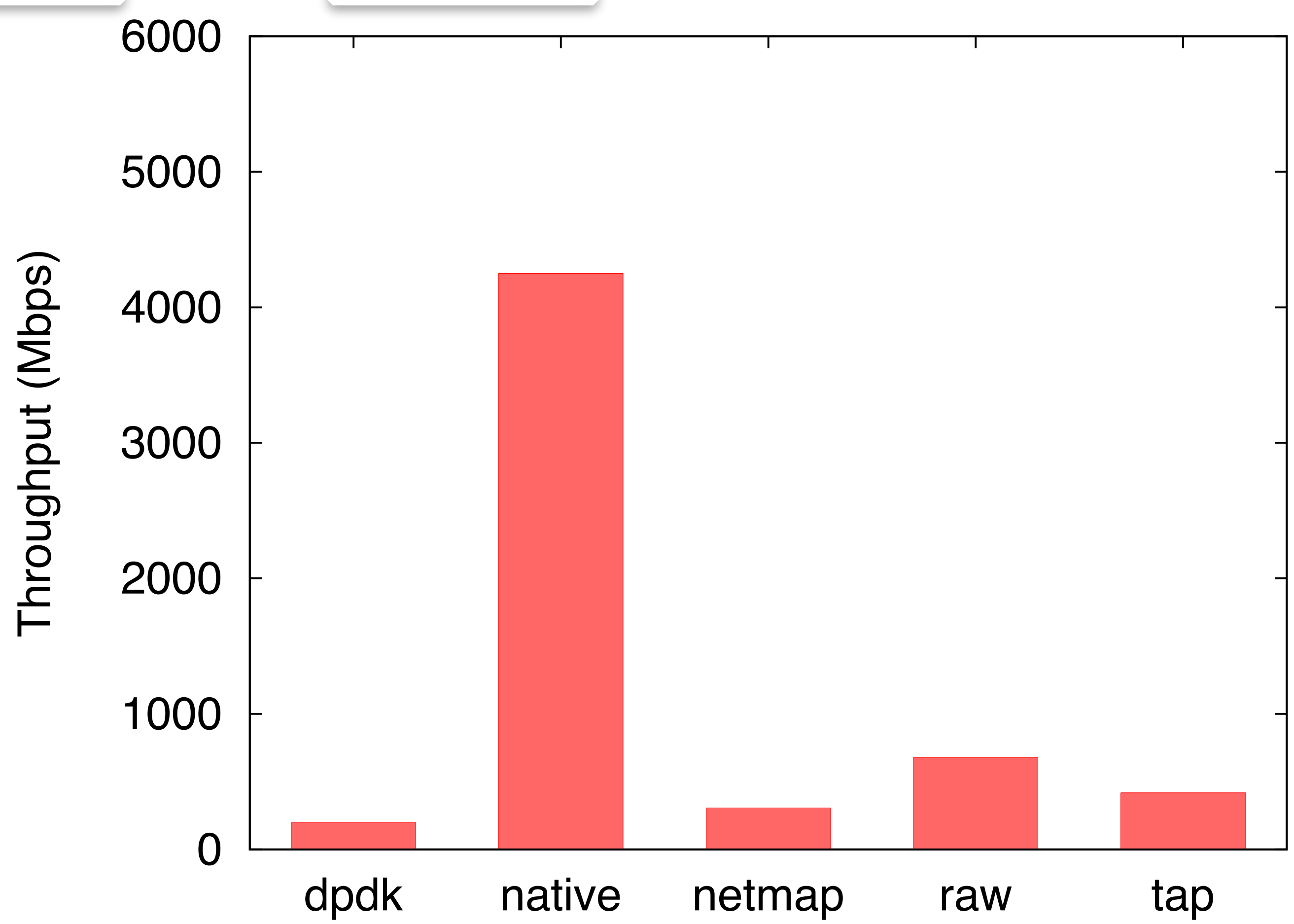
native: ping A.B.C.D
others: ./nuse ping A.B.C.D

L3 Routing

Sender->NUSE->Receiver



ping (RTT)



throughput (1024byte,UDP)

Alternatives

- UML/LKL (1proc/1vm, no POSIX i/f)
- Containers (can’t change kernel)
- scratch-based (mTCP,Mirage)
- rumpkernel (in NetBSD)

Limitations

- ad-hoc kernel glues required
  - when we changed a member of a struct, LibOS needs to follow it
- Performance drawbacks on NUSE
  - adapt known techniques (mTCP)

(not) Conclusions

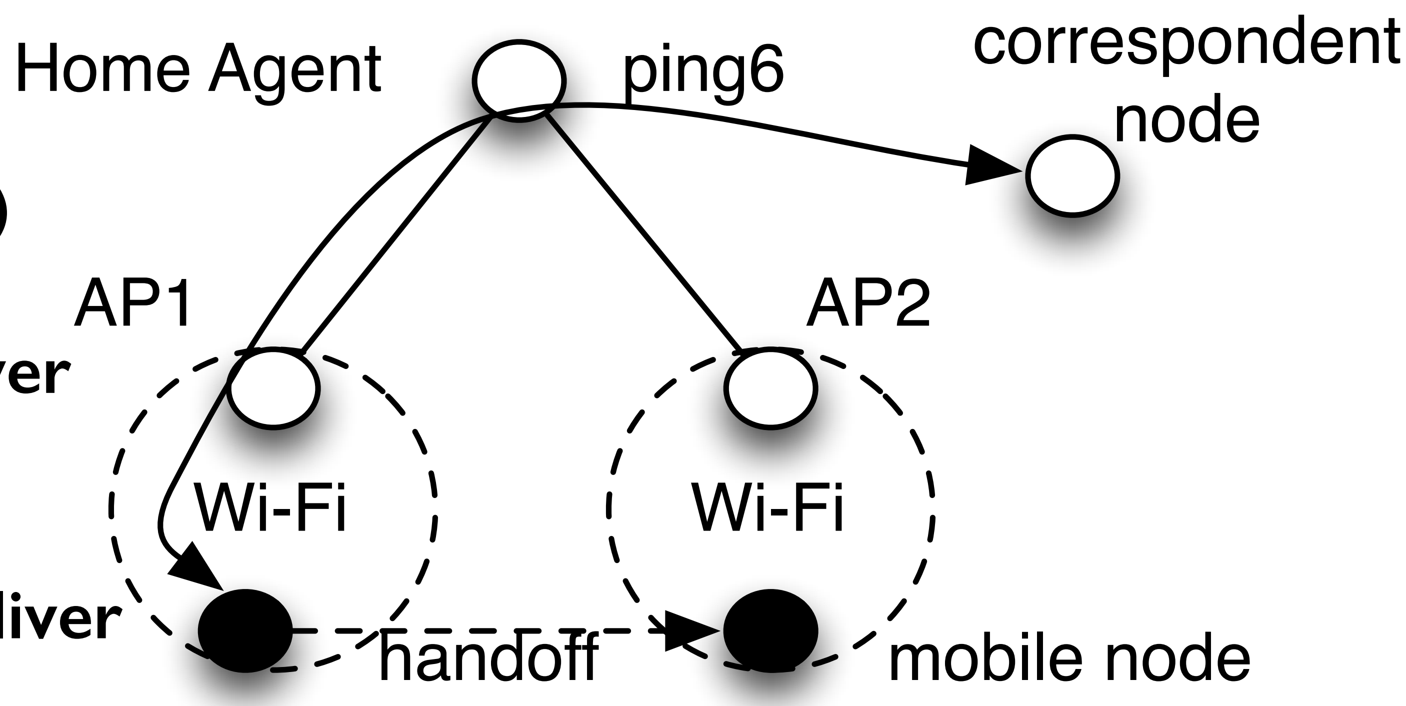
- An abstraction for multiple benefits
- Conservative
  - Use past decades effort as much
  - with a small amount of effort
- Planing to RFC for upstreaming

- github: https://github.com/libos-nuse/net-next-nuse
- DCE: http://bit.ly/ns-3-dce
- twitter: @thehajime

Backups

Bug reproducibility

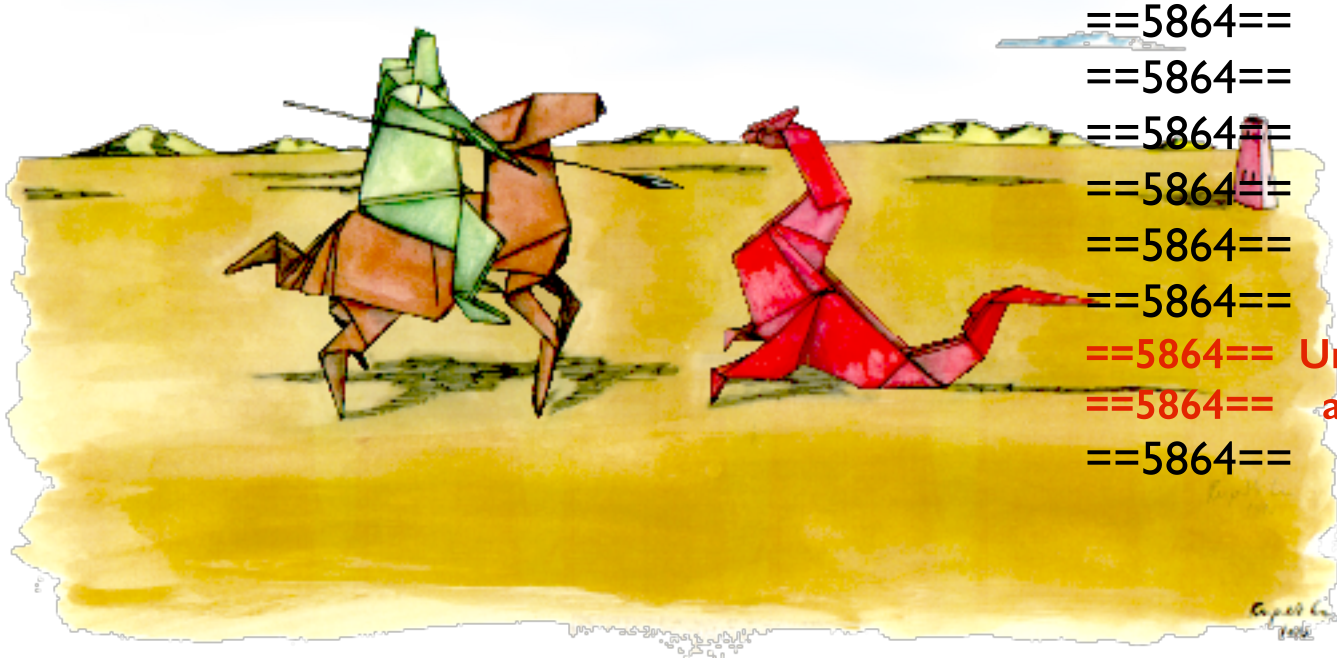
(gdb) b mip6_mh_filter if **dce_debug_nodeid()==0**
Breakpoint 1 at 0x7ffff287c569: file net/ipv6/mip6.c, line 88.
<continue>
(gdb) bt 4
#0 **mip6_mh_filter**
(sk=0x7ffff7f69e10, skb=0x7ffff7cde8b0)
at net/ipv6/mip6.c:109
#1 0x00007ffff2831418 in **ipv6_raw_deliver**
(skb=0x7ffff7cde8b0, nexthdr=135)
at net/ipv6/raw.c:199
#2 0x00007ffff2831697 in **raw6_local_deliver**
(skb=0x7ffff7cde8b0, nexthdr=135)
at net/ipv6/raw.c:232
#3 0x00007ffff27e6068 in **ip6_input_finish**
(skb=0x7ffff7cde8b0)
at net/ipv6/ip6_input.c:197



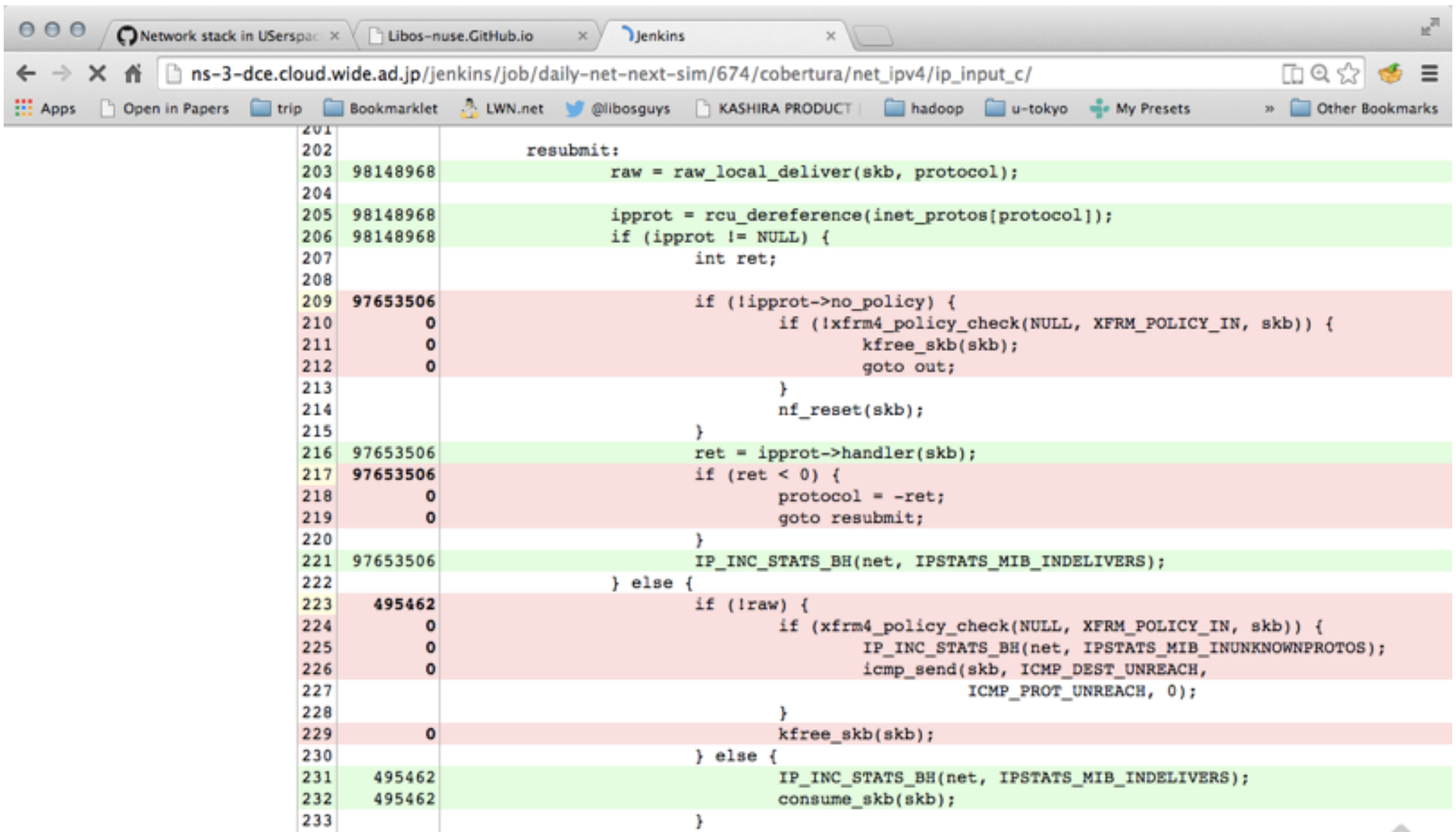
Debugging

- Memory error detection
  - among distributed nodes
  - in a single process
- using Valgrind

```
==5864== Memcheck, a memory error detector
==5864== Copyright (C) 2002-2009, and GNU GPL'd, by Julian Seward et al.
==5864== Using Valgrind-3.6.0.SVN and LibVEX; rerun with -h for copyright i
==5864== Command: ../build/bin/ns3test-dce-vdl --verbose
==5864==
==5864== Conditional jump or move depends on uninitialised value(s)
==5864==    at 0x7D5AE32: tcp_parse_options (tcp_input.c:3782)
==5864==    by 0x7D65DCB: tcp_check_req (tcp_minisocks.c:532)
==5864==    by 0x7D63B09: tcp_v4_hnd_req (tcp_ipv4.c:1496)
==5864==    by 0x7D63CB4: tcp_v4_do_rcv (tcp_ipv4.c:1576)
==5864==    by 0x7D6439C: tcp_v4_rcv (tcp_ipv4.c:1696)
==5864==    by 0x7D447CC: ip_local_deliver_finish (ip_input.c:226)
==5864==    by 0x7D442E4: ip_rcv_finish (dst.h:318)
==5864==    by 0x7D2313F: process_backlog (dev.c:3368)
==5864==    by 0x7D23455: net_rx_action (dev.c:3526)
==5864==    by 0x7CF2477: do_softirq (softirq.c:65)
==5864==    by 0x7CF2544: softirq_task_function (softirq.c:21)
==5864==    by 0x4FA2BE1: ns3::TaskManager::Trampoline(void*) (task-manag
==5864== Uninitialised value was created by a stack allocation
==5864==    at 0x7D65B30: tcp_check_req (tcp_minisocks.c:522)
==5864==
```



Fine-grained parameter coverage



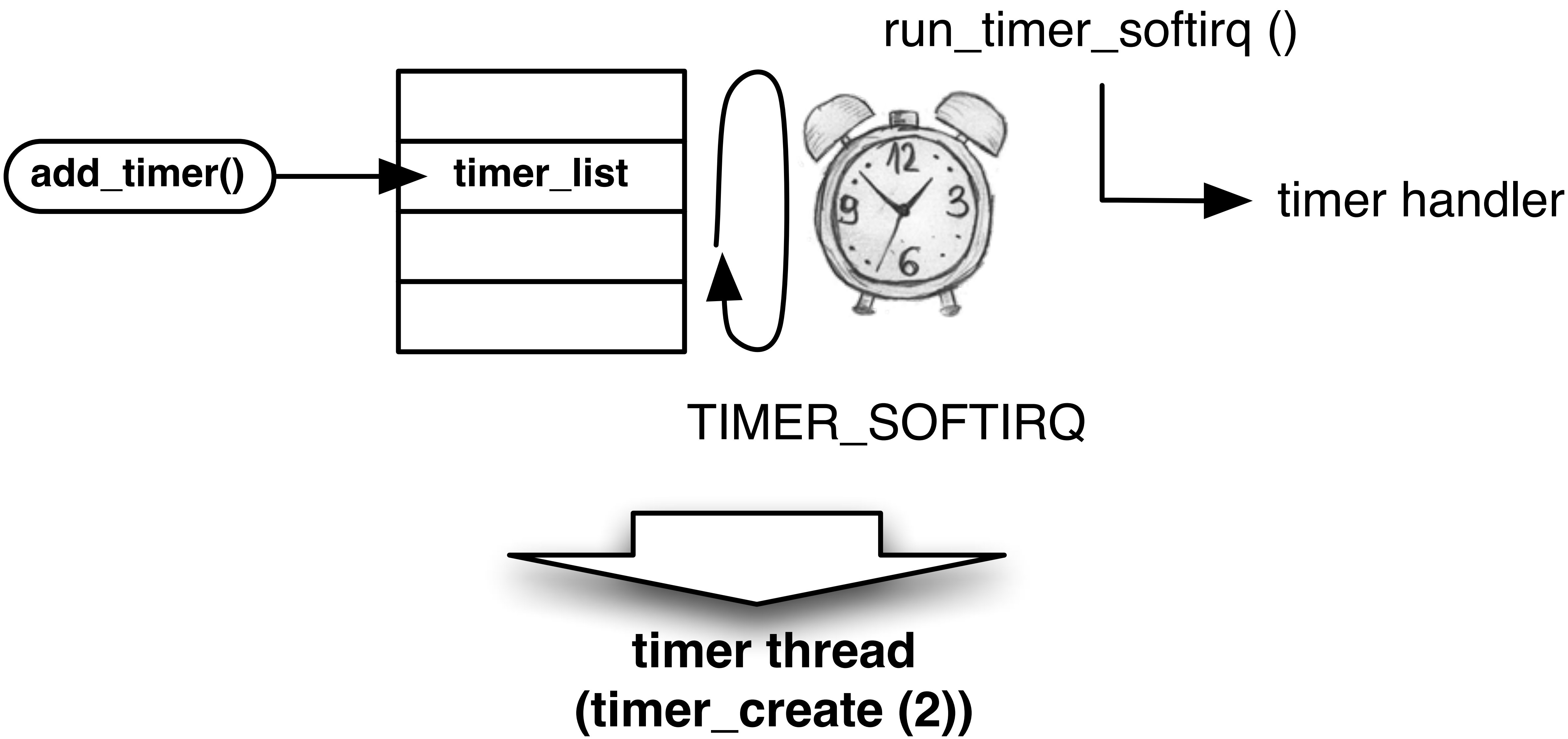
Code coverage measurement with DCE
With fine-grained network, node, protocol parameters

1) kernel build

- build kernel source tree w/ the patch
  - make menuconfig ARCH=sim
  - make library ARCH=sim

➔ libnuse-linux-3.17-rc1.so

Example: How timer works



Tx callgraph

sendmsg ()	(socket API)
lib_sock_sendmsg ()	(NUSE)
sock_sendmsg ()	
ip_send_skb ()	
ip_finish_output2 ()	
dst_neigh_output ()	(existing
neigh_resolve_output ()	-kernel)
arp_solicit ()	
dev_queue_xmit ()	
lib_dev_xmit ()	(NUSE)
nuse_vif_raw_write ()	

Rx callgraph

vNIC rx	start_thread () nuse_netdev_rx_trampoline () **nuse_vif_raw_read** () **lib_dev_rx** () netif_rx ()	(pthread) (NUSE) (ex-kernel)
softirq rx	start_thread () **do_softirq** () net_rx_action () process_backlog () __netif_receive_skb_core () ip_rcv ()	(pthread) (NUSE) (ex-kernel)