MoonGen
A Scriptable High-Speed Packet Generator
Design and Implementation

Paul Emmerich

January 30th, 2016
FOSDEM 2016

Chair for Network Architectures and Services
Department of Informatics
Technical University of Munich (TUM)
Source: www.spirent.com
MoonGen

- MoonGen is a software packet generator
- Cheaper than hardware boxes
- More flexible
- Key features
  - **Fast**: LuaJIT, DPDK\(^1\), explicit multi-core support
  - **Flexible**: Craft packets in real-time in user-defined Lua scripts
  - **Timestamping**: Utilize hardware features found on modern commodity NICs

\(^1\)Intel Dataplane Development Kit, [http://www.dpdk.org](http://www.dpdk.org)
Multi-threading in Lua

- No native support, some libraries exist
- Our solution: multiple independent LuaJIT VMs
- Maps to our problem domain
  - Generate different traffic flows in different threads
  - Inter-thread communication rarely needed
  - Modern NICs support multiple independent queues natively
  - Serialization (via Serpent\(^2\)) and C functions for (slow) inter-VM communication

\(^2\text{https://github.com/pkulchenko/serpent}\)
Architecture

MoonGen Core

- Userscript
- config API
- data API

DPDK

- NIC
- Port
- Q_0, ..., Q_n

Lua VM

- Userscript master
- Userscript slave
- spawn

config API

- data API

data API
Example: Generating Load

```lua
function loadSlave(queue)
    local mempool = memory.createMemPool(function(buf)
        buf:getUdpPacket():fill()
    end)
    local bufs = mempool:bufArray()
    while mg.running() do
        bufs:alloc(60)
        for i, buf in ipairs(bufs) do
            local pkt = buf:getUdpPacket()
            pkt.ip4.src:set(math.random(0, 2^32 - 1))
            pkt.udp.src:set(math.random(0, 2^16 - 1))
        end
        bufs:offloadUdpChecksums()
        queue:send(bufs)
    end
end
```
Packet data in Lua

- Different protocols are combined to build protocol stacks
- Endless combinations (e.g., tunnels in tunnels...)
- Efficient access to all protocol header fields required
Packet data in Lua

- Different protocols are combined to build protocol stacks
- Endless combinations (e.g., tunnels in tunnels…)
- Efficient access to all protocol header fields required

- Dynamically build LuaJIT FFI cdata structs

```lua
vxlanPkt = createPacket("eth", "ip4", "udp",
    "vxlan",{"eth", "innerEth"},{"ip4", "innerIp4"})
local pkt = vxlanPkt(buf)
pkt.innerIp4:setSrc("10.0.0.1")
```

- Dynamically create “class” for the whole protocol stack
- Extremely fast modification operations
- Memory layout defined by cdata struct (can be sent out directly)
Summary

- User-defined Lua scripts instead of configuration or DSLs
- LuaJIT is really really fast
- LuaJIT FFI cdata for packet structs
- ≥ 10 Gbit/s per CPU core (≥ 15 million packets per second)
- Execute user-defined script code for each packet
- LuaJIT FFI and C libraries for low-level stuff (drivers)
Q & A

Try MoonGen yourself!

https://github.com/emmericp/MoonGen

Questions?
[Backup slide] Performance I: Lua can be faster than C

- UDP packets from varying source IP addresses

![Graph showing CPU frequency vs. Packet rate (Mpps) for MoonGen and Pktgen-DPDK]

- Pktgen-DPDK needs a complicated main loop that covers all possibilities
- MoonGen can use a tight inner loop
Generate random UDP packets on 2 10 Gbit NICs
8 calls to Lua’s standard `math.random` per packet
CPUs artificially clocked down to 1.2 GHz
Generate random UDP packets on 2 10 Gbit NICs
8 calls to Lua’s standard `math.random` per packet
CPUs artificially clocked down to 1.2 GHz