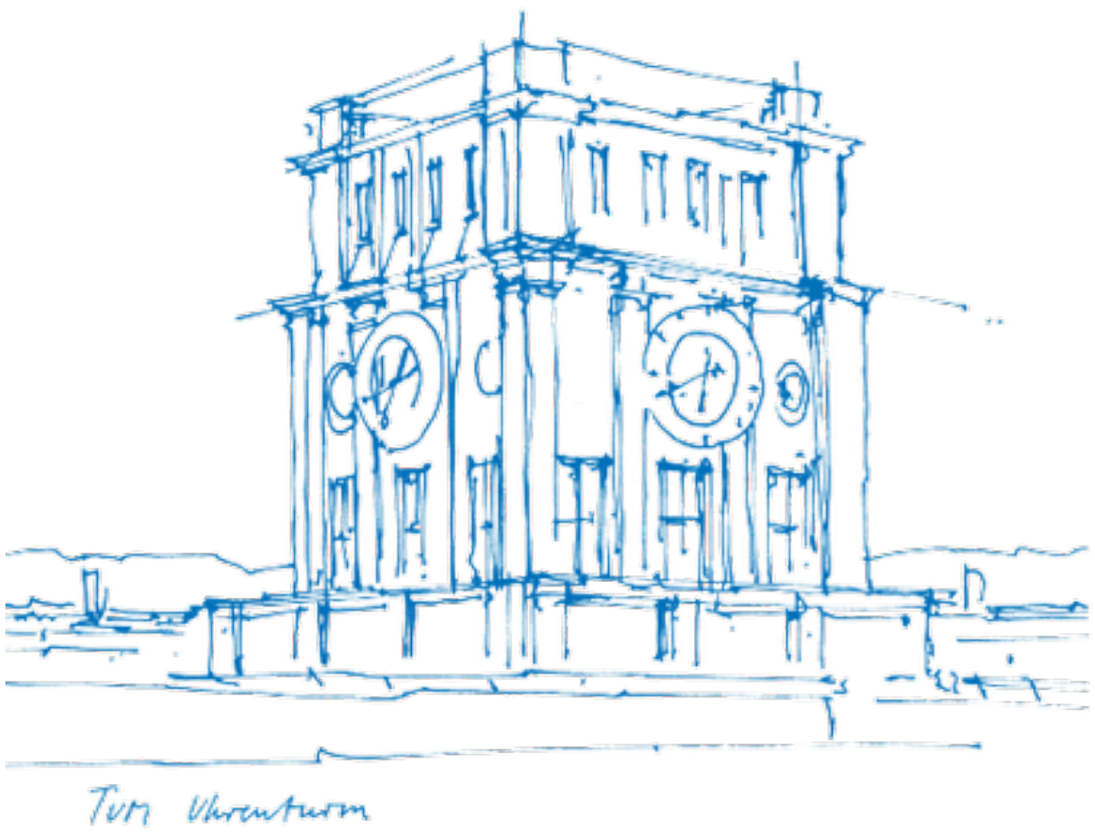
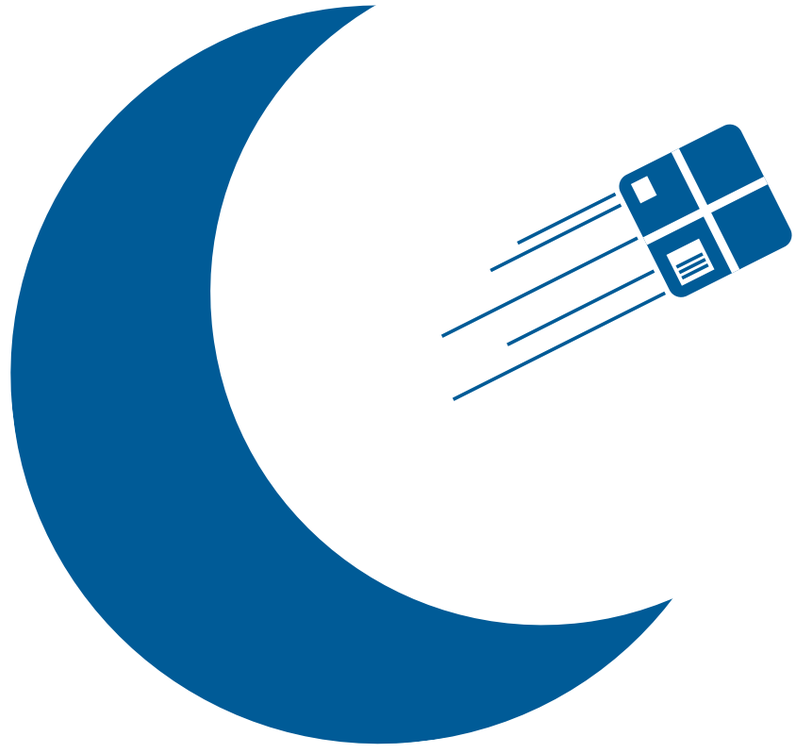
Rapid prototyping of DPDK applications with libmoon

Paul Emmerich
emmericp@net.in.tum.de
Technical University of Munich
Chair of Network Architectures and Services
DPDK Summit, 27.9.2017



About me

- PhD student at Technical University of Munich
- Started in 2014, DPDK user since 2013
- PhD thesis about testing network devices
- Built the MoonGen packet generator for this
  - Talked about MoonGen here last year
  - Often used in academia nowadays :)



https://github.com/emmericp/MoonGen

Paul Emmerich, Sebastian Gallenmüller, Daniel Raumer, Florian Wohlfart, and Georg Carle.
MoonGen: A Scriptable High-Speed Packet Generator. *Internet Measurement Conference (IMC) 2015*, October 2015.

Using DPDK in academia

- Lots of one-off prototypes or implementations deployed only once
- Proof-of-concepts, benchmarks, traffic analysis
- Work often “outsourced“ to student theses
  - Advisor for > 10 Bachelor’s and Master’s Theses using DPDK
- Teaching DPDK
  - Exercise for our Advanced Computer Networks lecture: build a router
  - Simple DPDK-based router, 4 VMs for each student for testing
  - ~35 students participated
  - Provided boilerplate code for device and memory initialization
  - Most common mistake: handling mbufs

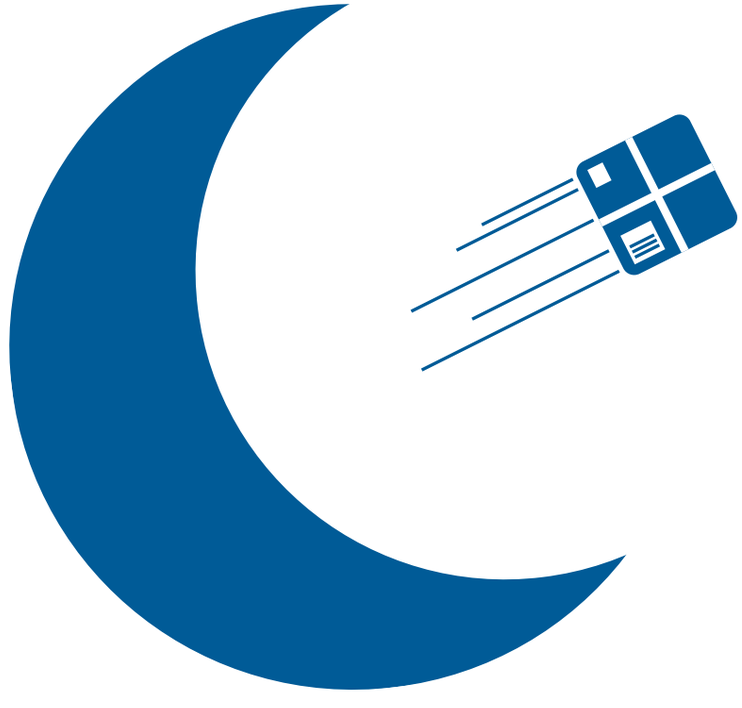
Problems with DPDK for prototypes

- Lots of boilerplate code required for initialization
- Things that should be simple often require lots of code
- Build system can be… annoying
- Hard to get students started

- Typical time frame for a student project: 4-6 months part-time
  - Need to understand and research the general topic
  - Design and implement a prototype
  - Often: brush up on C skills before
  - Analyze results, write thesis

- Hard to really get into DPDK in this scope for most students

The libmoon library



- libmoon is a Lua wrapper for DPDK
- Originally written for the MoonGen packet generator

- Why Lua?
  - Scripting language
  - Can call existing C/C++ functions without overhead
  - As fast as C/C++

- Comes with all the utilities you need for prototypes
  - Simplified device initialization with reasonable defaults
  - Command line parsing library
  - Predefined helper threads for statistics, ARP, ICMP, LACP, …
  - MoonGen packet library for structured access to packet data

Example: l2-forward.lua

- libmoon example script, similar to DPDK's l2fwd
  - Multi-threaded
  - Multi-queue with RSS
  - Prints statistics
- 40 lines of code (without comments)

- DPDK l2fwd: ~650 lines of code (without comments)
  - Used to be more in older versions, so it improved!

- We based our prototypes on the DPDK examples before libmoon
- Huge mess of copied & pasted code just to get basic functionality, e.g.:
  - IO statistics
  - Device configuration

Example: reflector.lua

- Reflects packets on multiple links, multi-thread/queue with RSS
- Worker thread, started once per queue pair

```
function reflector(rxQ, txQ)
    local bufs = memory.bufArray()
    while lm.running() do
        local rx = rxQ:tryRecv(bufs, 1000)
        for i = 1, rx do
            local pkt = bufs[i]:getEthernetPacket()
            local tmp = pkt.eth:getDst()
            pkt.eth:setDst(pkt.eth:getSrc())
            pkt.eth:setSrc(tmp)
            local vlan = bufs[i]:getVlan()
            if vlan then
                bufs[i]:setVlan(vlan)
            end
        end
        txQ:sendN(bufs, rx)
    end
end
```

Small things that can help a lot

fish /home/emmericp/libmoon — ssh • ssh slowpoke — ⌘1

```
[INFO]  Found 10 usable devices:
   Device 0: 24:8A:07:B0:27:AC (Mellanox Technologies MT27710 Family [ConnectX-4 Lx])
   Device 1: 24:8A:07:B0:27:AD (Mellanox Technologies MT27710 Family [ConnectX-4 Lx])
   Device 2: 90:E2:BA:71:7E:48 (Intel Corporation 82580 Gigabit Network Connection)
   Device 3: 90:E2:BA:71:7E:49 (Intel Corporation 82580 Gigabit Network Connection)
   Device 4: 90:E2:BA:71:7E:4A (Intel Corporation 82580 Gigabit Network Connection)
   Device 5: 90:E2:BA:71:7E:4B (Intel Corporation 82580 Gigabit Network Connection)
   Device 6: 68:05:CA:3A:A3:5C (Intel Corporation Ethernet Controller X710 for 10GbE SFP+)
   Device 7: 68:05:CA:3A:A3:5D (Intel Corporation Ethernet Controller X710 for 10GbE SFP+)
   Device 8: 0C:C4:7A:C4:66:2C (Intel Corporation Ethernet Controller 10G X550T)
   Device 9: 0C:C4:7A:C4:66:2D (Intel Corporation Ethernet Controller 10G X550T)
[INFO]  Waiting for devices to come up...
[INFO]  Device 6 (68:05:CA:3A:A3:5C) is up: 10000 MBit/s
[INFO]  Device 7 (68:05:CA:3A:A3:5D) is up: 10000 MBit/s
[INFO]  2 devices are up.
[Device: id=6] RX: 14.40 Mpps, 7372 Mbit/s (9676 Mbit/s with framing)
[Device: id=7] RX: 14.38 Mpps, 7364 Mbit/s (9665 Mbit/s with framing)
[Device: id=6] TX: 14.38 Mpps, 7364 Mbit/s (9666 Mbit/s with framing)
[Device: id=7] TX: 14.40 Mpps, 7373 Mbit/s (9678 Mbit/s with framing)
[Device: id=6] RX: 14.74 Mpps, 7547 Mbit/s (9906 Mbit/s with framing)
[Device: id=7] RX: 14.68 Mpps, 7514 Mbit/s (9862 Mbit/s with framing)
[Device: id=6] TX: 14.67 Mpps, 7514 Mbit/s (9862 Mbit/s with framing)
[Device: id=7] TX: 14.74 Mpps, 7547 Mbit/s (9906 Mbit/s with framing)
^C[Device: id=6] RX: 14.74 (StdDev nan) Mpps, 7547 (StdDev nan) Mbit/s (9906 Mbit/s with framing), total
37132137 packets with 2376456768 bytes (incl. CRC)
```

Check out libmoon on GitHub

libmoon comes with a lot of examples to get started

https://github.com/libmoon/libmoon

Questions?

reflector.lua main/setup

```
function master(args)
    local lacpQueues = {}
    for i, dev in ipairs(args.dev) do
        local dev = device.config{
            port = dev,
            rxQueues = args.threads + (args.lacp and 1 or 0),
            txQueues = args.threads + (args.lacp and 1 or 0),
            rssQueues = args.threads
        }
        -- last queue for lacp
        if args.lacp then
            table.insert(lacpQueues,
                {rxQueue = dev:getRxQueue(args.threads), txQueue = dev:getTxQueue(args.threads)})
        end
        args.dev[i] = dev
    end
    device.waitForLinks()

    -- setup lacp if requested
    if args.lacp then
        lacp.startLacpTask("bond0", lacpQueues)
        lacp.waitForLink("bond0")
    end

    -- print statistics
    stats.startStatsTask{devices = args.dev}

    for i, dev in ipairs(args.dev) do
        for i = 1, args.threads do
            lm.startTask("reflector", dev:getRxQueue(i - 1), dev:getTxQueue(i - 1))
        end
    end
    lm.waitForTasks()
end
```

reflector.lua boilerplate and CLI

```
local lm     = require "libmoon"
local memory = require "memory"
local device = require "device"
local stats  = require "stats"
local lacp   = require „proto.lacp"

function configure(parser)
    parser:argument("dev", "Devices to use."):args("+"):convert(tonumber)
    parser:option("-t –threads",
       "Number of threads per   device."):args(1):convert(tonumber):default(1)
    parser:flag("-l --lacp", "Try to setup an LACP channel.")
end
```