Network stack personality in Android phone

Cristina Opriceana, **Hajime Tazaki** (IIJ Research Lab.)

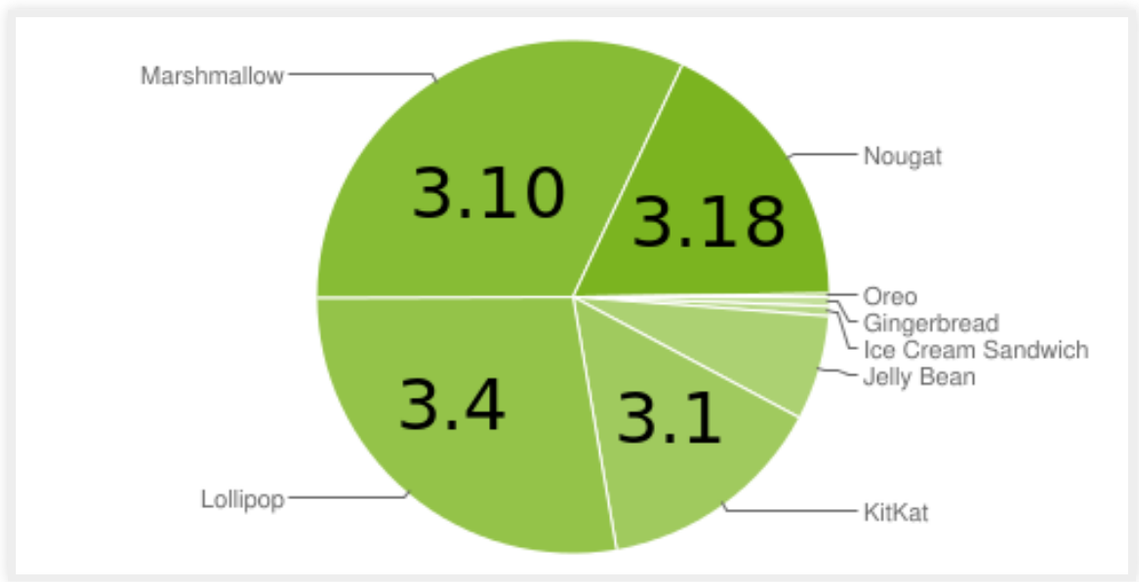
Linux netdev 2.2, Seoul, Korea

07 Nov. 2017

Android

a platform of billions devices

- billions installed Linux kernel
- slow delivery of software updates
- Questions
  - When our upstreamed code available ?
  - What if I come up with a great protocol ?



https://developer.android.com/about/dashboards/index.html

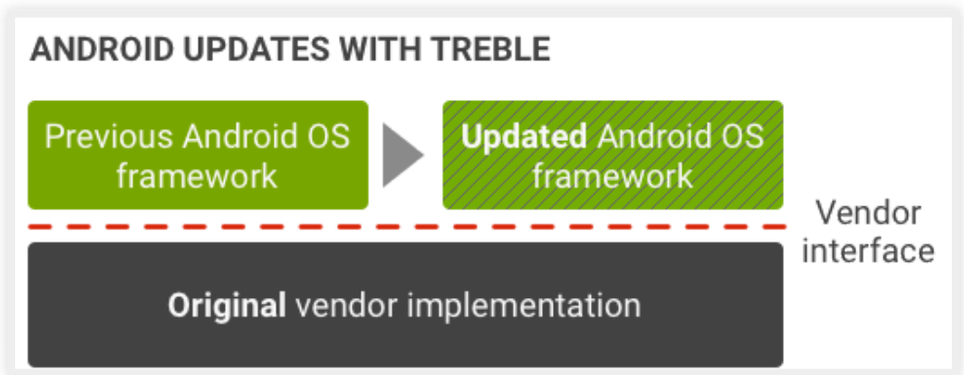
Android (cont'd)

- When our upstreamed code available ?
  - wait until base kernel is upgraded
  - backport specific function
- What if I come up with a great protocol ?
  - craft your own kernel and put into your image

Looong delivery to all billions devices

Approaches to alleviate the issue

- Virtualization (KVM on Android)
  - Overhead isn't negligible to embedded devices
- Project Treble (since Android O)
  - More modular platform implementation
- Fushia
  - Rewrite OS from scratch



https://source.android.com/devices/architecture/treble

An alternate approach

- network stack personality
  - use own network stack implemented in userspace
  - no need to replace host kernels
  - but (try to) preserve the application compatibility
- NUSE (network stack in userspace)
  - No delay of network stack update
  - Application can choose a network stack if needed

Userspace implementations

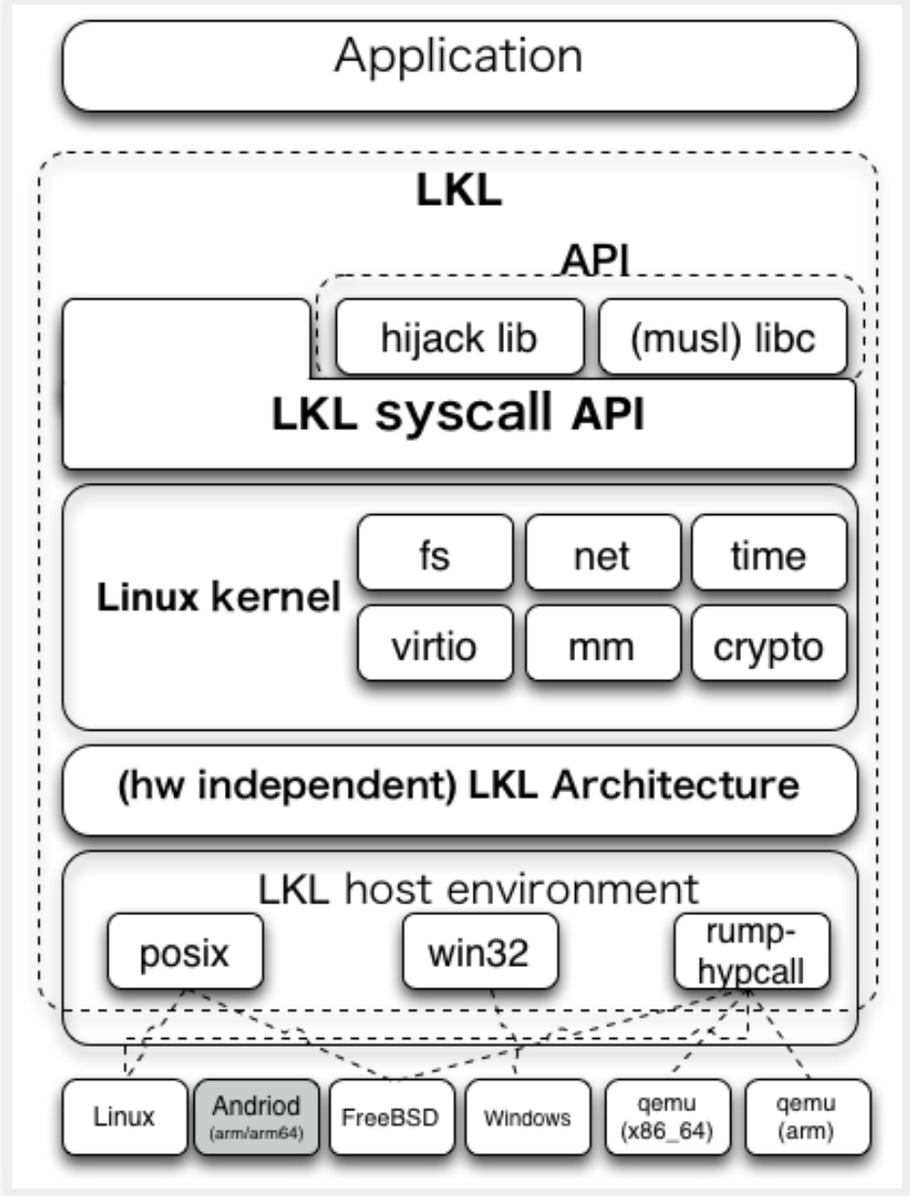
- Toys, Misguided People
- Selfish

- Don't care whether toy or not
- Do care whether it's useful or not

(TODO: replace image)

Linux Kernel Library intro (again)

- Out-of-tree architecture (h/w-independent)
- Run Linux code on various ways
  - with a reusable library
- h/w dependent layer
  - on Linux/Windows /FreeBSD uspace, unikernel, on UEFI,
  - (ns-3)
  - **Android**



LKL: current status

- Sent RFC (Nov. 2015)
  - no update on LKML since then
- have evolved a lot
  - various extensions (offload, SMP, json config, unikernel, etc)

https://github.com/lkl/linux

Extensions to LKL

- Android (arm/arm64) support (lkl/linux#372)
- raw socket extension (only handle ETH_P_IP) (not yet)
- hijack library enhance (not yet)
- HOWTO

```
% LD_PRELOAD=liblkl-hijack.so netperf XXX # console app
% setprop wrap.app LD_PRELOAD=liblkl-hijack.so # Java app
```

hijack library

- For smooth replacement for Android UI app (java-based)
  - bionic is more familiar than glibc ?
  - only socket-related calls are redirected
  - mixture of host and lkl descriptors (can avoid)

TODO (a fig of hijack)

New feature introduction

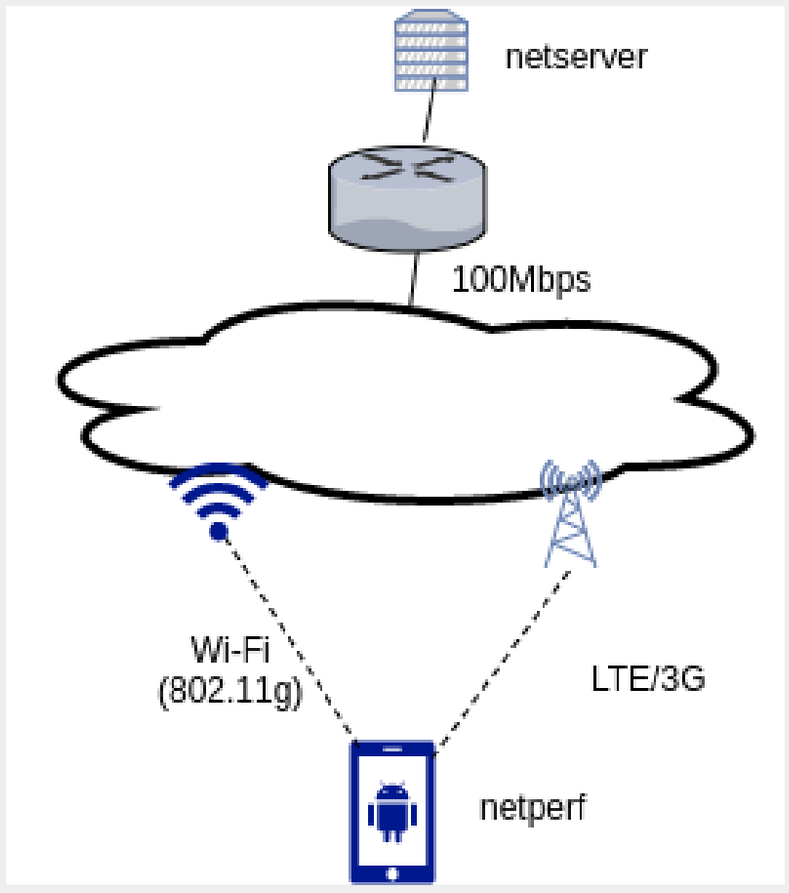
- Example
  - Multipath TCP (http://multipath-tcp.org/)
  - out-of-tree for long time
  - verify site (cat /proc/net/mptcp base detection) http://amiusingmptcp.de/

Demo

No penalty with userspace network stack ?

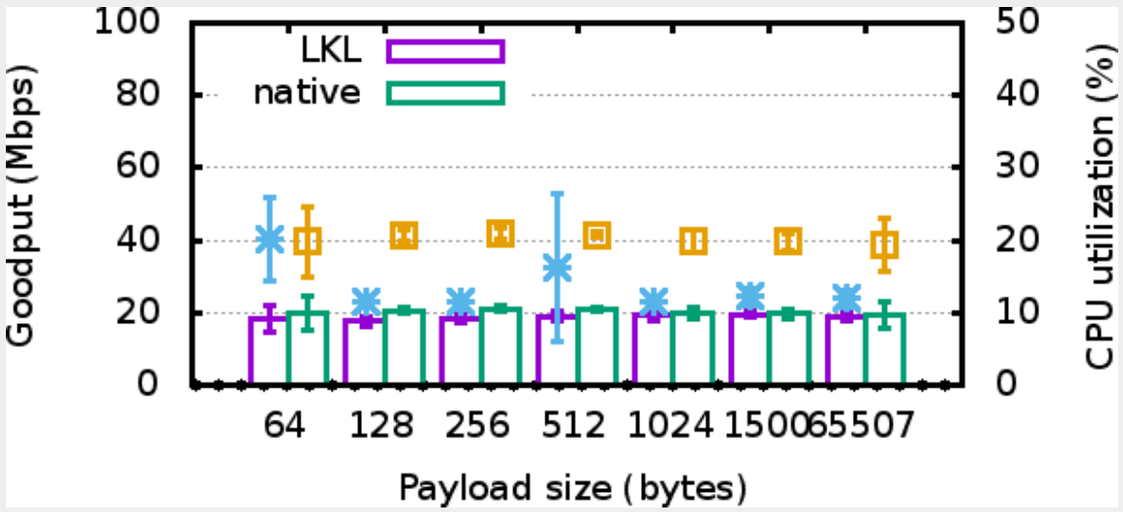
- Condition
  - To use Linux mptcp w/o replacing kernel
  - With tolerable amount of overhead
- Questions
  - Is NUSE working fine (Will users wanna use it) ?
  - How different from native Linux kernel ?

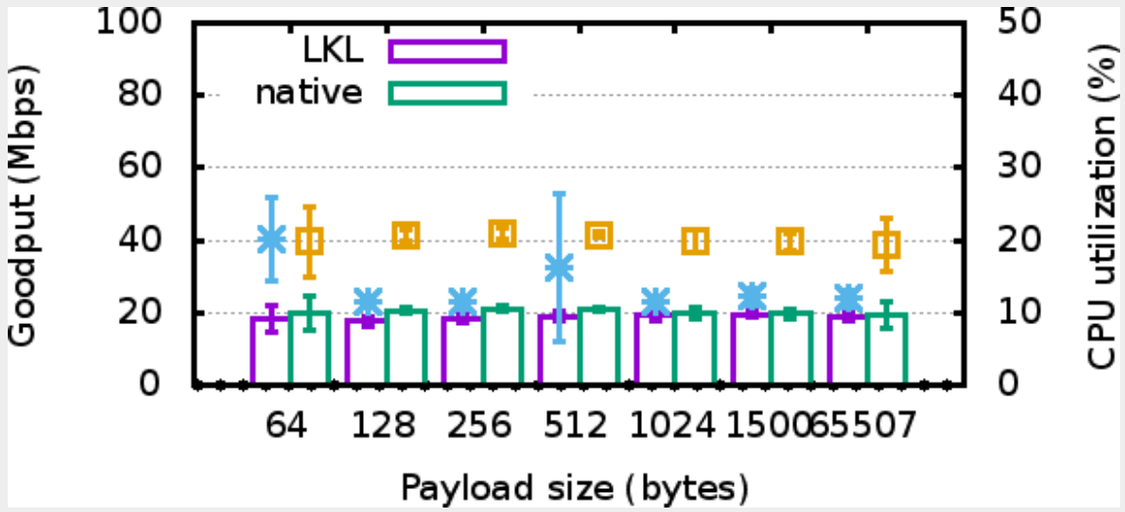
netperf measurement



- Client
  - Nexus5 anrdoid 6.01 (rooted)
  - LTE, wifi
  - LKL arm/android patched
  - or native kernel
- Server
  - Ubuntu 16.04 (amd64) on KVM
  - virtio/Etherlink (uplink: 100 Mbps)
  - mptcp-4.4.70 (v0.92)
- Software
  - netperf 2.7.x
  - 10 seconds TCP_STREAM, TCP_MAERTS
  - 5 trials, over 64-64K byte packet

Single path (Wi-Fi only, 9/14)





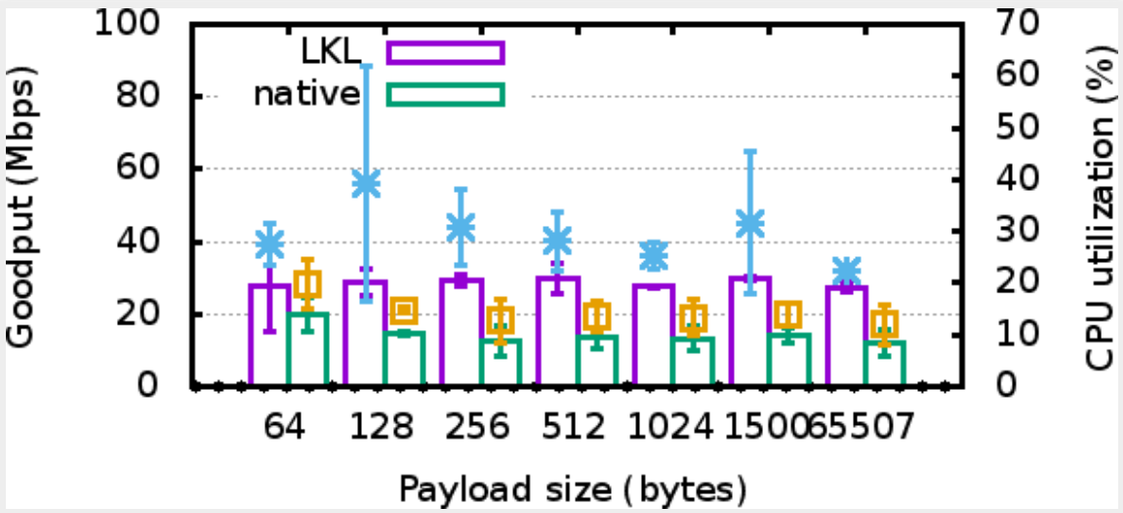
Tx (TCP_STREAM)

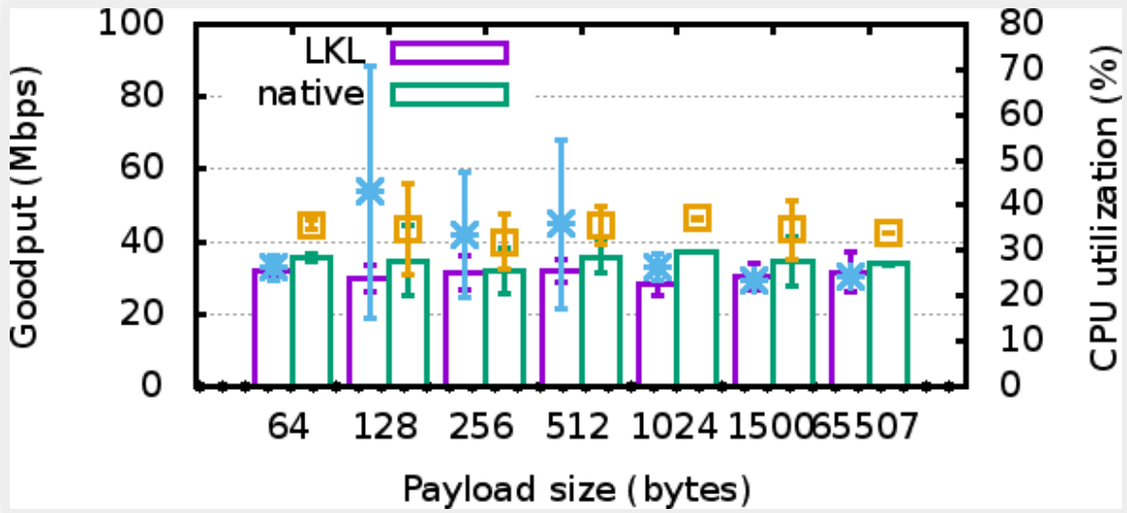
- Condition
  - phone: LKL v.s. native kernel

Rx (TCP_MAERTS)

- Comparable goodput
- CPU utilization: LKL < native

Multipath TCP





Tx (TCP_STREAM)

- Condition
  - phone: LKL v.s. mptcp kernel

Rx (TCP_MAERTS)

- Tx with native kernel offers less goodput
  - even it's using multipath
- results are unstable

Multipath TCP (Korea/KT)

(TBA)

- Condition
  - phone: LKL v.s. mptcp kernel
- (TODO) To be measured

Observation

- IP conflicts may heavier
  - processed twice (host/lkl) per packet
- Results are always unstable
  - difficult measurement under wireless media

TODO (a fig of overhead)

Limitations

- Implementations
  - DHCP only boot time (handover will fail)
  - IPv4 only on cellular interface (rmnet0)
- Required tweaks
  - grant NET_RAW permission (packet socket)
  - need filter out RST packet from host

```
iptables -A OUTPUT -p tcp --tcp-flags RST RST  -j DROP
```

Further investigations

- profiling
- other platform (iOS11 shipped userspace implementation)

Summary

- Use out-of-tree kernel as a library on Android
  - make your code easier to distribute
  - But with privileged installation/operation
- Comparable performance wireless communication