Networking APIs

Breakout at

Munich Internet Research Retreat Raitenhaslach (MIR^3) 2017

(Lars Eggert, Michio Honda, Jörg Ott, Hagen Paul Pfeifer, Marius Strobl, Florian Westphal, Lars Wischhof)

Assumption 1: Standard Internet Architecture

In the discussion, the currently deployed Internet architecture is assumed.

i.e. new networking paradigms – that would require completely new functionalities of the Networking-API - are **out-of-scope**, e.g.:

- Information Centric / Content Based Networking
- Delay-Tolerant Networking
- Vehicular Networking
- …

Assumption 2: IoT Requirements out-of-scope

Specific requirements of IoT are not considered in the discussion since

- The IoT device very specific requirements, i.e. regarding energy-efficiency, so that the Socket API is not applicable/is not being used, or
- Energy-consumption of current Socket API is not a problem because sending consumes much more energy than inefficient implementation of Socket API
  (e.g. copying data from userland to kernelspace)

Problems of the current API (1/3)

The 1970s/1980s Socket API does not scale anymore!

- Large number of connections, e.g. thousands of TCP connections
  ➔ large number of file-descriptors, inefficient
- Semantic: once you read, memory is yours

Work-arounds in order to cope with this inefficiency, e.g.:

- sendfile (directly copying from file desc. to socket desc. without copying data to user-space)
- sendmessage with zero-copying / zero-copy sockets (page re-mapping between kernel/user-space)

Problems of the current API (2/3)

Importance of hardware-offloading has increased:

- What functions should be performed in hardware?
  - Checksum calculation, etc.
  - Complete networking stack?
    (feasible but many disadvantages: bugfixes, improvements, …)
  - Interface to the hardware?

Problems of the current API (3/3)

Current API is fine for most standard cases but becomes a bottleneck for high performance applications:

- Copying data, no packet-oriented processing of data in application (application sends stream of data, transport layer needs segments)

Again, workarounds possible, e.g.

- Fast packet processing in userland
- StackMap + netmap framework (dedicated NIC for one application, etc.)

Desireable Properties

- Isolation of network-stack and application
  (main danger is not breaking the system but access to data that the application is not allowed to access)
- Energy efficiency (➔ mobile)
- And of course: high efficiency/performance (➔ data center)

Solution Ideas

- Dedicated I/O CPUs
- Packetized processing of data in the application
- Integration of GPU and networking (offloading on GPU)
- Reduce overhead in kernel, e.g. avoid queuing of TCP ACK packets (but: requires changes in driver!)
- Reduce overhead of system call

??? New API or "just" solving performance issues of existing API ???