ePICS

Engineering Proprioception in Computing Systems

Efficient Implementation of Dynamic Protocol Stacks

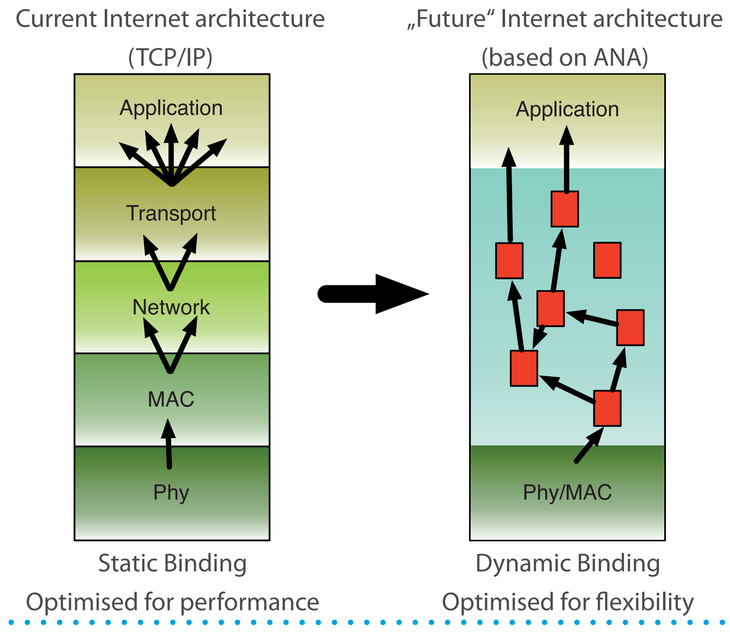
What are Dynamic Protocol Stacks and why are they Needed?

Vision: computing everywhere
- Small, inexpensive, robust networked devices
- Distributed at all scales throughout everyday life

Challenges:
- Heterogeneous nodes
- Geographical distribution to build situation awareness
- Embedded into non-technical context

Observed first steps:
- Penetration of mobile devices (Smart Phones, Tablet PCs, etc.)
- Deployed sensor networks (e.g., in dikes or permafrost regions)
- Video surveillance networks

Challenges for a Network Architecture in Pervasive Computing
- Flexibility with respect to functionality and mobility
- Resource Limitations
- TCP/IP not suitable
  - Fixed infrastructure
  - No resource limitation
- Clean slate network architecture
  - Meta architecture
  - Divide networking functionality into functional blocks (FBs)
  - Build protocol stack that is optimised for the current situation
  - Adapt protocol stack to changing environment on the fly

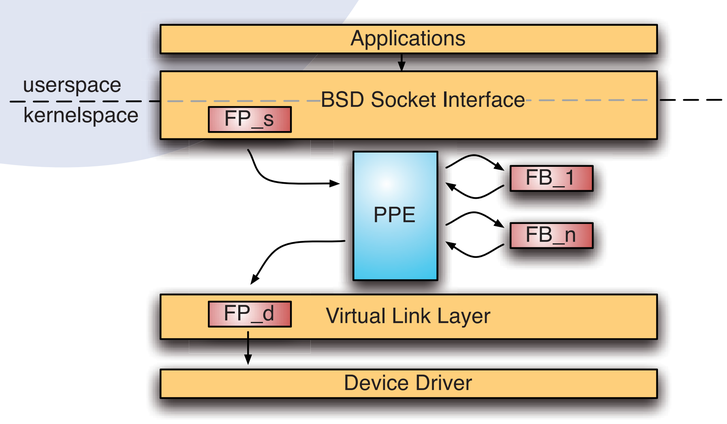


How can Dynamic Protocol Stacks be Implemented?

Basics
The LANA network system is built similarly to the network subsystem of the Linux kernel. Applications can send and transmit packets via the BSD socket interface, packets are processed in the kernel space and transmitted over / received from a device driver.

Modules
Each functional block is implemented as a Linux kernel module. Upon module insertion a constructor for the creation of an instance of the functional block is registered with the LANA core. Functional blocks can either drop a packet, forward a packet to either ingress or egress direction or duplicate a packet. After having processed a packet the functional block returns the identifier of the next functional block that should process this packet.

Packet Processing Engine (PPE)
The PPE is responsible for calling one functional block after the other and for queuing packets that need to be processed.

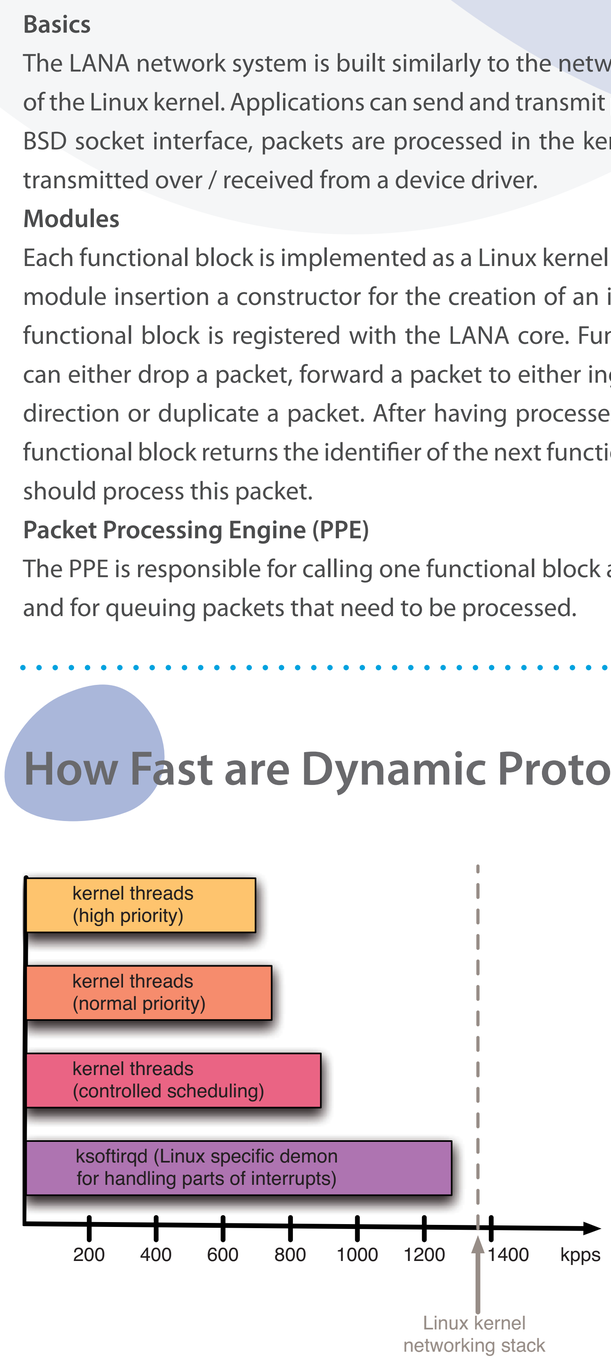


Virtual Link Layer
The hardware and device driver interfaces are hidden from the PPE behind a virtual link interface, which allows for a simple integration of different underlaying networking technologies such as Ethernet, Bluetooth or InfiniBand.

Configuration
- `add, rm`
  Adds (removes) an FB from the list of available FBs in the kernel.
- `set`
  sets properties of an FB with a key=value semantic.
- `bind, unbind`
  Binds (unbinds) an FB to another FB in order to be able to send messages to it.
- `replace`
  Replaces one FB with another FB. The connections between the blocks are maintained. Private data can either be transferred to the new block or dropped.

Currently Available Blocks
- Functionality: Ethernet, Berkeley Packet Filter
- Packet Flow: Tee, Forwarding, Sink (packet counter)

How Fast are Dynamic Protocol Stacks?



Maximum packet reception rate for
- 64 Byte packets
- Intel core 2 Quad Q6600 2.40 GHz
- 4GB Ram
- Intel 82566DC-2 Nic
- Linux 3.0rc1

If the PPE is executed as a dedicated kernel thread the ksoftirq daemon, which initially receives the packets, suffers from starvation. Changing priorities or controlling scheduling does not significantly improve the performance. Only executing the PPE in ksoftirqd context achieves a performance similar to the Linux kernel networking stack (configured to drop any packet after reception).

What brings the Future?

Performance Evaluation of Real Scenarios
We will compare the performance of real scenarios in LANA with the performance in other systems (e.g., default Linux stack, Click router, etc.).

Autonomous Configuration
We will work on mechanisms that automatically configure protocol stacks based on the applications as well as the networks needs.

Hardware Acceleration
We plan to enhance this system with support for hardware accelerators. The hardware support will be FPGA based in order to allow for the runtime re-configuration of the protocol stack. The adaptation of the hardware will be done by partial reconfiguration of the FPGA. For the communication between software and hardware the ReconOS operating system will be used.

Software
The current software is available under the GNU General Public License from http://repo.or.cz/w/ana-net.git
Interested researchers are encouraged to download and try the software and report their feedback to us.

Authors and Acknowledgement

Ariane Keller: ariane.keller@tik.ee.ethz.ch
Daniel Borkmann: dborkma@tik.ee.ethz.ch
Wolfgang Mühlbauer: muehlbauer@tik.ee.ethz.ch
ETH Zurich, Switzerland

The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement nº257906.
www.epics-project.eu

SEVENTH FRAMEWORK PROGRAMME