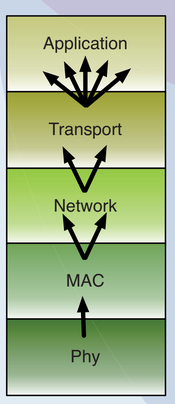
ePICS

Engineering Proprioception in Computing Systems

Efficient Implementation of Dynamic Protocol Stacks

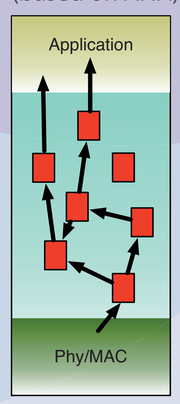
What are Dynamic Protocol Stacks and why are they needed?

Current Internet architecture



Static Binding

Dynamic Protocol Stack (based on ANA)



Dynamic Binding

Dynamic Protocol Stacks

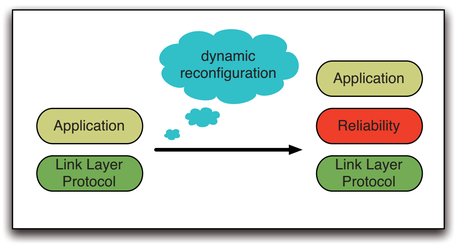
- Does not rely on TCP/IP architecture
- Does not imply any protocols to be used -> meta architecture
- Divide networking functionality into functional blocks (FBs)
- Adapt protocol stack at runtime to current requirements

Example Scenario 1: Sensor Network

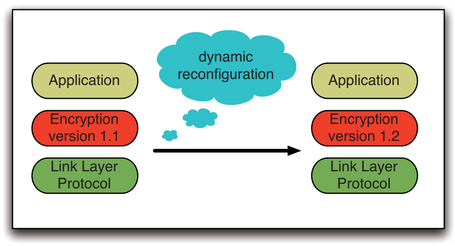
- No routing required -> no IP required
- Small amount of data sent -> tcp's complexity not required
- Network conditions change frequently

Example Scenario 2: Video Surveillance System

- 100% uptime required
- Data protection consideration might change
- Implementation bugs need to be fiexed



Sensor Network Scenario



Video Surveillance Scenario

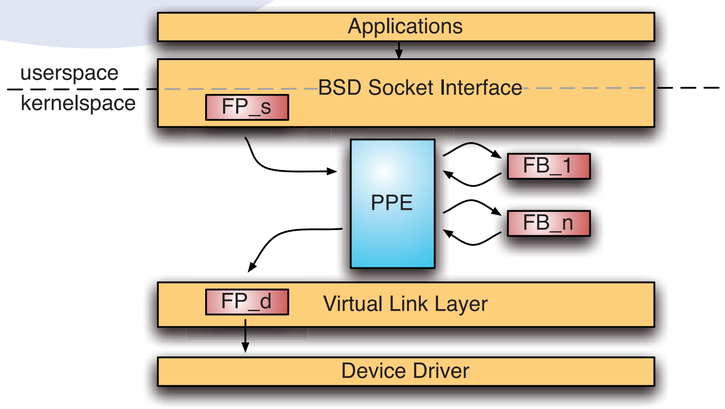
How can Dynamic Protocol Stacks be implemented?

Existing Approaches

FreeBSD Netgraph: dynamic protocol configuration but only up to IP
Click: configuration upon initialization, designed for routers
ANA: full flexibility but only proof of concept implementation

Our Approach

- Implementation in the Linux kernel space. Each functional block is a kernel module.
- Instances of the modules can be generated from user space.
- Packets can be dropped, forwarded to ingress or egress, duplicated or generated.
- Packet Processing Engine (PPE) controls the execution of the functional blocks. After finishing its task, each block returns control to the PPE.
- Virtual Link Layer hides hardware and device driver peculiarities in order to allow an easy extension to other transmission channels such as Bluetooth or InfiniBand.



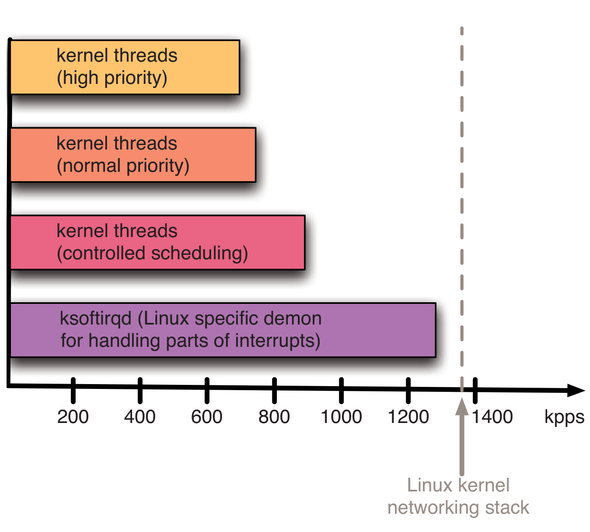
Configuration

- `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

Results

- Low level implementation influences the performance heavily.
- Interrupt processing and thread scheduling significantly influence the maximum number of packets that can be received.

What will the future bring?

Performance Evaluation of Real Scenarios

We will compare the performance of real scenarios in LANA with their 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 Acknowledgements

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

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement n°257906.
www.epics-project.eu

SEVENTH FRAMEWORK PROGRAMME