LARN
Latency- and Resilience-Aware Networking

Latency- and Resilience-Aware Networking (LARN)

SPP 1914 “Cyber-Physical Networking” Kickoff - Presentation

Prof. Dr.-Ing. Thorsten Herfet
Telecommunications Lab
Saarland Informatics Campus (SIC) Saarbrücken

Prof. Dr.-Ing. habil. Wolfgang Schröder-Preikschat
Distributed Systems and Operating Systems
Friedrich-Alexander-Universität (FAU) Erlangen-Nürnberg

October 21, 2016

Cyber-Physical Systems (CPS)

- ... provide a cross-cutting foundation framework to enable novel services.
  - Autonomous Vehicles
  - Smart Energy
  - Smart Cities
  - Smart Anything
- ... are required to provide **safe**, **secure** and **dependable** services.
- ... are **inherently interconnected** (Internet of Things, Cloud/Fog Computing, Industry 4.0, ...) leading to **Cyber-Physical Networks (CPN)**.
- To make CPNs safe and dependable, **latency and resilience requirements** have to be taken into account.

Cyber-Physical Systems (CPS)

- ... provide a cross-cutting foundation framework to enable novel services.
  - Autonomous Vehicles
  - Smart Energy
  - Smart Cities
  - Smart Anything
- ... are required to provide **safe**, **secure** and **dependable** services.
- ... are **inherently interconnected** (Internet of Things, Cloud/Fog Computing, Industry 4.0, ...) leading to **Cyber-Physical Networks (CPN)**.
- To make CPNs safe and dependable, **latency and resilience requirements** have to be taken into account.

CPS require different approaches to networking and operating systems

LARN
Latency- and Resilience-Aware Networking

Latency-Awareness

- Latency Avoiding and Hiding
- Bounded Execution Time
- Parallel Execution
- Preparatory Operations
- ...

Resilience-Awareness

- Environment Influences
- Hardware Failures
- Software Problems
- Situation-Dependent Adaptations
- ...

Latency-Awareness

- Latency Avoiding and Hiding
- Bounded Execution Time
- Parallel Execution
- Preparatory Operations
- ...

Resilience-Awareness

- Environment Influences
- Hardware Failures
- Software Problems
- Situation-Dependent Adaptations
- ...

Both needed together to provide a strong foundation for applications

Automated Repeat reQuest (ARQ)

- Reactively add redundancy.
- Retransmit after timeout.
- Ideal: Low RTT.

Forward Error Correction (FEC)

- Proactively add redundancy.
- Accumulate data and encode.
- Ideal: High RTT or multicast.

Automated Repeat reQuest (ARQ)

- Reactively add redundancy.
- Retransmit after timeout.
- Ideal: Low RTT.

Forward Error Correction (FEC)

- Proactively add redundancy.
- Accumulate data and encode.
- Ideal: High RTT or multicast.

Error control adds resilience, but increases latency

LARN
Latency- and Resilience-Aware Networking

Automated Repeat reQuest (ARQ)

- Reactively add redundancy.
- Retransmit after timeout.
- Ideal: Low RTT.

Forward Error Correction (FEC)

- Proactively add redundancy.
- Accumulate data and encode.
- Ideal: High RTT or multicast.

Error control adds resilience, but increases latency

Solution for Arbitrary Channels

Apply Adaptive Hybrid Error Correction (AHEC).

- Hybrid: FEC and ARQ at the same time.
- Adaptive: Incorporate...
  - ... channel parameters (latency, loss, maximum throughput) and
  - ... application requirements (maximum latency, tolerable residual loss, throughput).

Noisy-Channel Coding Theorem (Shannon 1945)

For a channel capacity

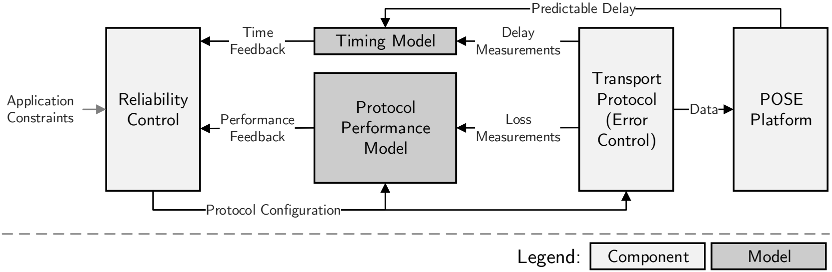
$$C_{Shannon} = \sup I(X;Y)$$

a transmission with rate $R < C$ and error probability $p_e \leq \epsilon$ is possible.

Finite Blocklength Channel Coding Rate (Polyanskiy et al. 2010)

For an error probability ϵ, channel capacity $C_{Shannon}$, blocklength N, channel dispersion V and complementary Gaussian cumulative distribution function Q, the maximal data rate is:

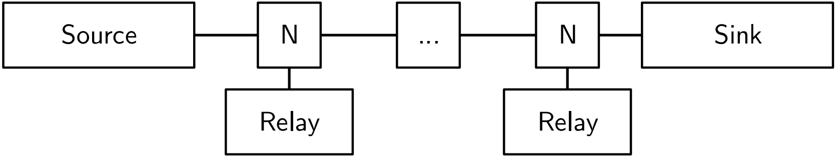
$$C_{Finite} = C_{Shannon} - \sqrt{\frac{V}{N}} \cdot Q^{-1}(\epsilon)$$



PRRT

- Transport-layer protocol taking resilience and latency into account.
- Works on any underlying system (e.g. Linux) and channel (wired or wireless).

Bounds on processing time when running on predictable platform?
Can soft guarantuees be provided?



TTS

- Network segments heterogeneous (varying loss and delay parameters).
- Coding parametrization depending on link parameters.
- Segmenting transmissions allows to fine-tune coding.
- Network functions (error, congestion, flow control) working end-to-end.

Segmentation (where? how many?) not trivial.

Baseline

- On loss-free, low-jitter paths TTS is worse than E2E.
- E2E performing better as TTS in 53% of measured cases.
- Why? TTS adds overhead in processing.

Reordering

- High jitter scenarios worsen performance of TTS.
- Why? Relays reinforce order.

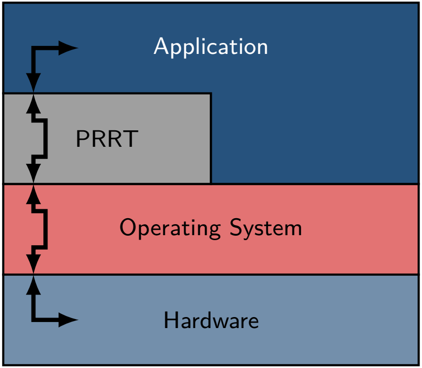
Error Control

- With loss, TTS is **nearly always** better than E2E.
- Mean and jitter reduce (4x less).
- Why? Retransmissions happened locally. Lost ACKs do not trigger unnecessary retransmits.

Flow Control

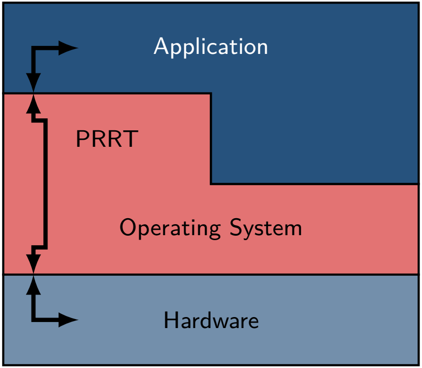
To-be-evaluated (in process).

Transparent Transmission Segmentation with TCP
Andreas Schmidt, Thorsten Herfet (ICCE-Berlin'16, NetCPS'16)



Problems

- OS $\rightarrow$ latency, jitter
- Unnecessary indirections
- Unpredictable hardware

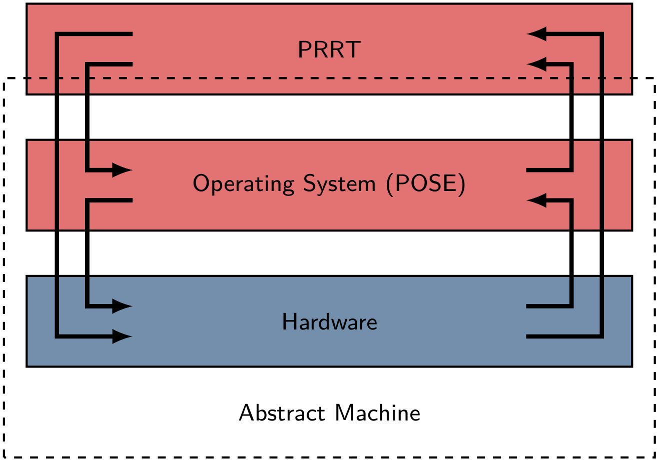


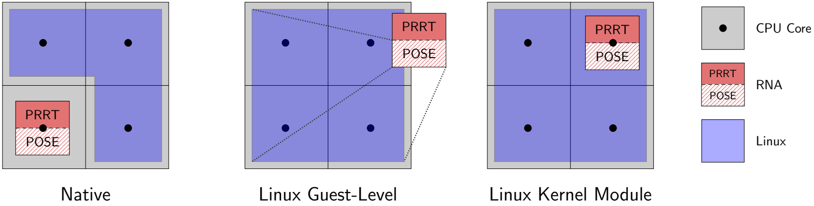
Problems

- OS → latency, jitter
- Unnecessary indirections
- Unpredictable hardware

Challenges

- Minimise latency, jitter
- Optimise data and control flow
- Tame hardware


PRRT
Operating System (POSE)
Hardware
Abstract Machine



Portability

- Target platforms: x86, ARM, ...
- Hosted and native environments
- Embedded to Multicore systems

Linux Compatibility

- Hybrid operating system
- Transparent to application code
- Transparent to network interface

Latency-Aware Process Management

- Maximise predictability
- Minimise latency where possible
- Hide latency where necessary

Latency-Aware Inter-Process Communication (IPC)

- Vertical: Cross-layer Communication
- Horizontal: Intra-Protocol Coordination

Communication Concepts

Minimal Base

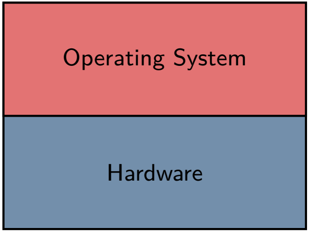
Guarded Sections: Structuring Aid for Wait-Free Synchronisation
Gabor Drescher, Wolfgang Schröder-Preikschat (ISORC 2015)

Hardware Feature Exploitation

- Maximise efficiency
- Minimise noise
- Eliminate unnecessary abstraction

Flyweight Resource Management

- Application aware strategies
- Speculative pre-allocation



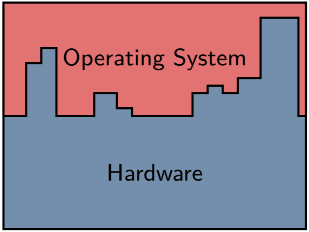
Sloth: Threads as Interrupts
Wanja Hofer, Daniel Lohmann, Fabian Scheler, Wolfgang Schröder-Preikschat (RTSS 2009)

Hardware Feature Exploitation

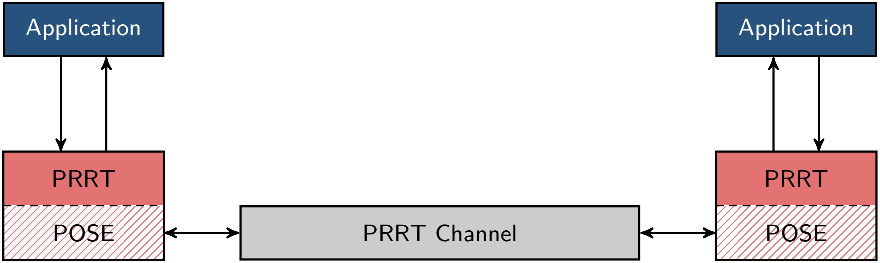
- Maximise efficiency
- Minimise noise
- Eliminate unnecessary abstraction

Flyweight Resource Management

- Application aware strategies
- Speculative pre-allocation



Sloth: Threads as Interrupts
Wanja Hofer, Daniel Lohmann, Fabian Scheler, Wolfgang Schröder-Preikschat (RTSS 2009)



Reliable Networking Atom (RNA)

- RNA = PRRT + POSE
- Single communication stack
- Provided in **two versions**

Pure software	For test scenarios, evaluation, prototyping.
Soft- and hardware	Realistic timing analysis, proper bounds on execution time.

- Interface: As simple to use as a **UDP socket**

Features

- RNA will be provided as a platform to other projects in the SPP.
- (Hardware) and libraries will be distributed.
- Enables projects focussing on control to use this infrastructure.

Preliminary Roadmap

- October 2017: RNA v0.1
  Working prototype: Integrating network and operating stacks.
- October 2018: RNA v0.9
  Improved prototype: Integrated hard- and software.
- July 2019: RNA v1.0
  Final polished version.

LARN

- Latency- and resilience must be considered at the same time.
- **PRRT** provides a network stack to guarantee both and approach channel limits.
- **POSE** minimises latency and jitter at system level.
- Both components will be provided to the project in form of **RNA**.

LARN

- Latency- and resilience must be considered at the same time.
- **PRRT** provides a network stack to guarantee both and approach channel limits.
- **POSE** minimises latency and jitter at system level.
- Both components will be provided to the project in form of **RNA**.

Thank you for your attention. Questions?