LARN
Latency- and Resilience-Aware Networking

Latency- and Resilience-Aware Networking

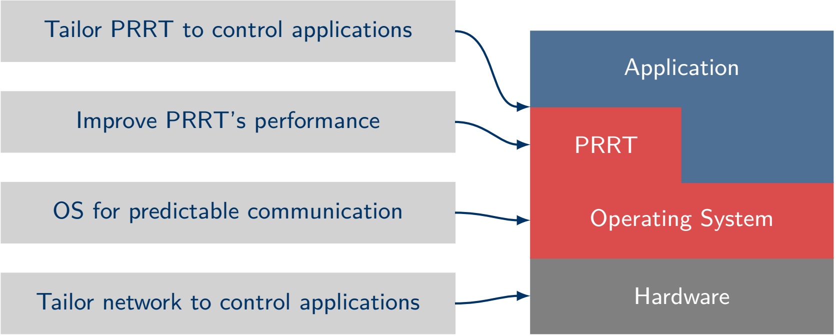
SPP 1914: "Cyber-Physical Networking"

http://larn.systems

Andreas Schmidt, Pablo Gil Pereira, Thorsten Herfet
Telecommunications Lab
Saarland Informatics Campus - Saarbrücken

Stefan Reif, Timo Hönig, Wolfgang Schröder-Preikschat
Department of Computer Science 4 (Distributed Systems and Operating Systems)
Friedrich-Alexander-Universität Erlangen-Nürnberg

April 23rd, 2018


Tailor PRRT to control applications
Improve PRRT's performance
OS for predictable communication
Tailor network to control applications
Application
PRRT
Operating System
Hardware

Software, Hardware & Algorithms

PRRT Predictable Reliable Real-time Transport protocol

- Packet Loss Measurement / Estimation
- BBR-based Congestion Control & Bottleneck Bandwidth Estimation
- Cross-layer Pacing between Application and Network
- API: Ordered Receive Modes, many other improvements

LARN

Software, Hardware & Algorithms

PRRT Predictable Reliable Real-time Transport protocol

- Packet Loss Measurement / Estimation
- BBR-based Congestion Control & Bottleneck Bandwidth Estimation
- Cross-layer Pacing between Application and Network
- API: Ordered Receive Modes, many other improvements

X-Lap Cross-Layer Timing Analysis

- Automated Detection of Causes of Latency & Jitter
- Automated Control-Flow Graph Extraction
- Energy Evaluations

LARN

Software, Hardware & Algorithms

PRRT Predictable Reliable Real-time Transport protocol

- Packet Loss Measurement / Estimation
- BBR-based Congestion Control & Bottleneck Bandwidth Estimation
- Cross-layer Pacing between Application and Network
- API: Ordered Receive Modes, many other improvements

X-Lap Cross-Layer Timing Analysis

- Automated Detection of Causes of Latency & Jitter
- Automated Control-Flow Graph Extraction
- Energy Evaluations

RNA Reliable Networking Atom

- Autonomous-Driving Car Scenario (BarCamp II)
- Wireless Embedded Real-Time Video Streaming Experiments

LARN

Definitions

- **Pace** := Time required to apply a certain step to a certain unit of data.
  (e.g. propagation time per packet or sampling time per sensor reading)
- A (sub-)system implements **pacing** iff it ensures that each step is executed at a pace that considers the bottleneck pace in the overall system's chain of processing steps.

Definitions

- **Pace** := Time required to apply a certain step to a certain unit of data.
  (e.g. propagation time per packet or sampling time per sensor reading)
- A (sub-)system implements **pacing** iff it ensures that each step is executed at a pace that considers the bottleneck pace in the overall system's chain of processing steps.

Invariant: Minimum pace in a system defines overall "throughput" of a system.

Definitions

- **Pace** := Time required to apply a certain step to a certain unit of data.
  (e.g. propagation time per packet or sampling time per sensor reading)
- A (sub-)system implements **pacing** iff it ensures that each step is executed at a pace that considers the bottleneck pace in the overall system's chain of processing steps.

Invariant: Minimum pace in a system defines overall "throughput" of a system.

Cross-Layer Pacing

- Measure paces of all layers: network, application, system.

LARN

Definitions

- **Pace** := Time required to apply a certain step to a certain unit of data. (e.g. propagation time per packet or sampling time per sensor reading)
- A (sub-)system implements **pacing** iff it ensures that each step is executed at a pace that considers the bottleneck pace in the overall system's chain of processing steps.

Invariant: Minimum pace in a system defines overall "throughput" of a system.

Cross-Layer Pacing

- Measure paces of all layers: network, application, system.
- Synchronise all the paces of all layers to the minimal pace.

Definitions

- **Pace** := Time required to apply a certain step to a certain unit of data.
  (e.g. propagation time per packet or sampling time per sensor reading)
- A (sub-)system implements **pacing** iff it ensures that each step is executed at a pace that considers the bottleneck pace in the overall system's chain of processing steps.

Invariant: Minimum pace in a system defines overall "throughput" of a system.

Cross-Layer Pacing

- Measure paces of all layers: network, application, system.
- Synchronise all the paces of all layers to the minimal pace.
- ... rinse, repeat.

Near Zero Queuing Delay

- Network bottleneck bandwidth measured: Minimal bound on interval between packets of given size.
- If adhering to this interval, packets arrive at either an idle link or an empty buffer.

Near Zero Queuing Delay

- Network bottleneck bandwidth measured: Minimal bound on interval between packets of given size.
- If adhering to this interval, packets arrive at either an idle link or an empty buffer.

Just-In Time Processing

- Application pace allows to anticipate a `send()` call.
- Preparatory tasks, e.g. memory allocation, thread wakeups, can be done in prior.

LARN

Near Zero Queuing Delay

- Network bottleneck bandwidth measured: Minimal bound on interval between packets of given size.
- If adhering to this interval, packets arrive at either an idle link or an empty buffer.

Just-In Time Processing

- Application pace allows to anticipate a `send()` call.
- Preparatory tasks, e.g. memory allocation, thread wakeups, can be done in prior.

Reduced "Waste"

- The operating system and physical computing platform can run exactly at the speed of application and transport layer.
- This allows to reduce clock-cycles, by avoiding polling, or slow-down the processor to prolong the lifetime of battery-driven devices.

Background

- Channel limited by *BtlBw*, which can be measured using BBR [Google'16].
- Application sends packets of size L and with frequency f (data rate $R_{app} = L \cdot f$).

Cross-Layer Pacing ensures $BtlBw \equiv R_{app}$ by controlling f.

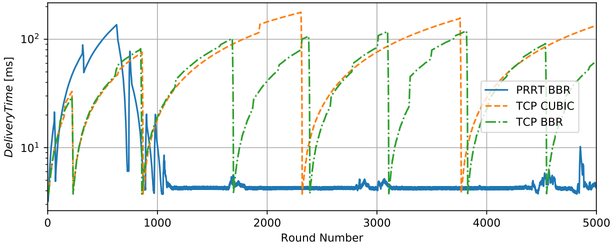
Background

- Channel limited by $BtlBw$, which can be measured using BBR [Google'16].
- Application sends packets of size L and with frequency f (data rate $R_{app} = L \cdot f$).

Cross-Layer Pacing ensures $BtlBw \equiv R_{app}$ by controlling f.

Implementation

- The BBR algorithm paces packets to the bottleneck bandwidth.
- The application is allowed to place one packet in the socket and the next `send()` call block until the packet is sent to the wire (after pacing period has passed).
- Alternatives:
  - The application can query the socket for the bottleneck and adjust its sampling rate or sensor resolution.
  - The operating system "slows down" the application.

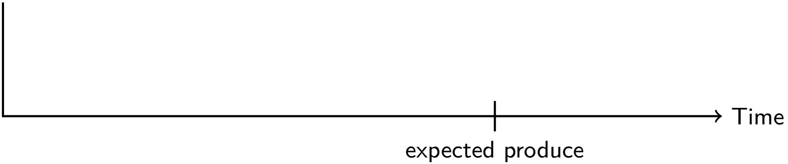


Results

- Delivery times within $1 - 4\times$ the propagation delay for most of the cases.
- PRRT achieves near-zero queueing after startup phase.
- In spite of limited send buffers, TCP cannot achieve this.

PRRT + BBR we can effectively pace an application to the speed of the network.

So far, we only synchronise the pace of the **application** and the **network**.



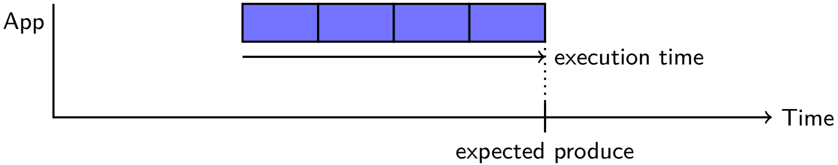
Operating System

- Pacing-aware scheduling.
- Energy-efficient pacing.

Hardware

- Measure maximum pace.
- Synchronise node-local and global paces (CPU throttling).

So far, we only synchronise the pace of the **application** and the **network**.



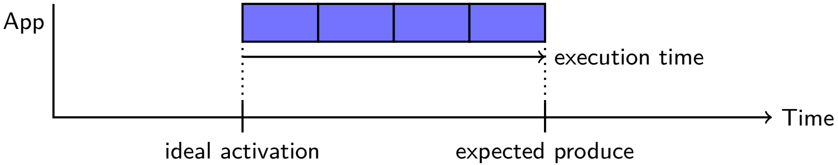
Operating System

- Pacing-aware scheduling.
- Energy-efficient pacing.

Hardware

- Measure maximum pace.
- Synchronise node-local and global paces (CPU throttling).

So far, we only synchronise the pace of the **application** and the **network**.



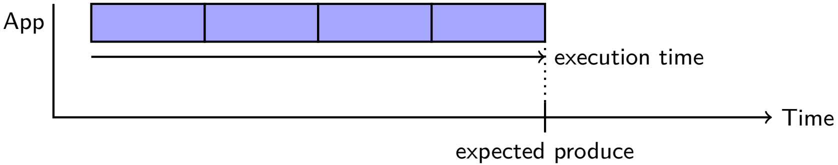
Operating System

- Pacing-aware scheduling.
- Energy-efficient pacing.

Hardware

- Measure maximum pace.
- Synchronise node-local and global paces (CPU throttling).

So far, we only synchronise the pace of the **application** and the **network**.



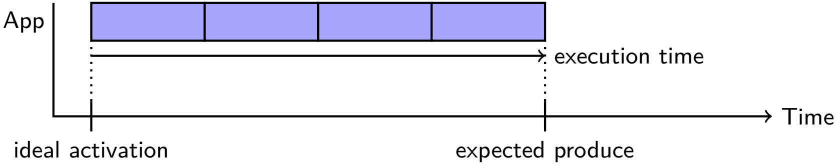
Operating System

- Pacing-aware scheduling.
- Energy-efficient pacing.

Hardware

- Measure maximum pace.
- Synchronise node-local and global paces (CPU throttling).

So far, we only synchronise the pace of the **application** and the **network**.



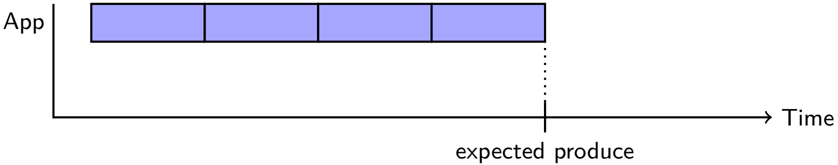
Operating System

- Pacing-aware scheduling.
- Energy-efficient pacing.

Hardware

- Measure maximum pace.
- Synchronise node-local and global paces (CPU throttling).

So far, we only synchronise the pace of the **application** and the **network**.



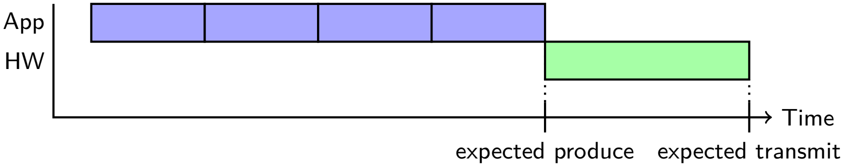
Operating System

- Pacing-aware scheduling.
- Energy-efficient pacing.

Hardware

- Measure maximum pace.
- Synchronise node-local and global paces (CPU throttling).

So far, we only synchronise the pace of the **application** and the **network**.



Operating System

- Pacing-aware scheduling.
- Energy-efficient pacing.

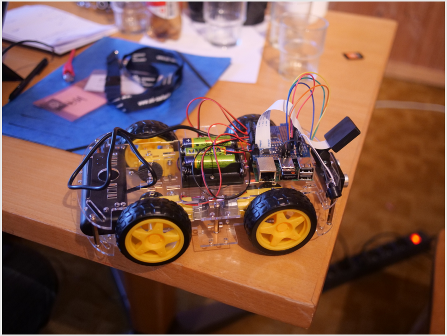
Hardware

- Measure maximum pace.
- Synchronise node-local and global paces (CPU throttling).

LARN

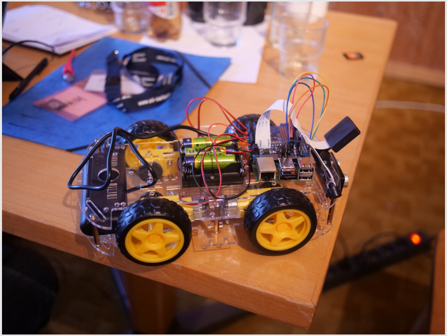
System

- Raspberry Pi 3 (w/ 802.11n)
- Pi Camera
- Chassis and Motor HAT
- Ultrasonic Sensors



System

- Raspberry Pi 3 (w/ 802.11n)
- Pi Camera
- Chassis and Motor HAT
- Ultrasonic Sensors



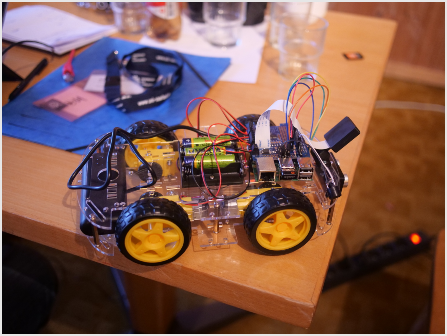
Line-Following ("Autonomous Driving")

- Camera captures line and transmits video via PRRT.
- Edge controller extracts line, determines angle, and determines control outputs.
- Target speed transmitted back and applied on the motor.

Edge2Car Communication

System

- Raspberry Pi 3 (w/ 802.11n)
- Pi Camera
- Chassis and Motor HAT
- Ultrasonic Sensors



Line-Following ("Autonomous Driving")

- Camera captures line and transmits video via PRRT.
- Edge controller extracts line, determines angle, and determines control outputs.
- Target speed transmitted back and applied on the motor.

Edge2Car Communication

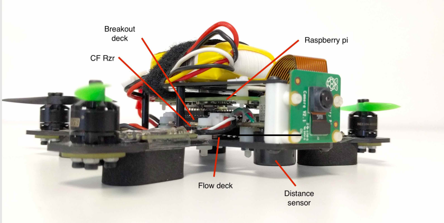
Car-Following ("Platooning")

- First car follows line.
- Second car follows but keeps distance to first car.

Car2Car Communication

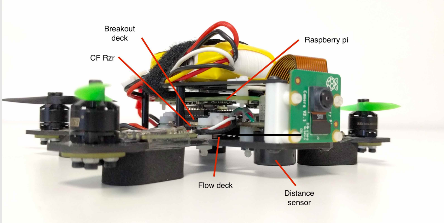
System

- Pi Zero W + Pi Camera
- CrazyFlie (from BitCraze.io)
  - Optical flow sensor (X,Y position).
  - Laser-based ranging (Z position).



System

- Pi Zero W + Pi Camera
- CrazyFlie (from BitCraze.io)
  - Optical flow sensor (X,Y position).
  - Laser-based ranging (Z position).



1. Mobile Real-time Video Streaming

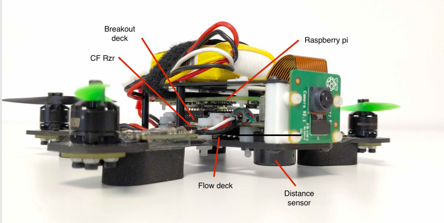
- PHY/MAC: 802.11n
- NET: IP
- TRANS: PRRT

Goal: Reliable & Timely Video Stream

LARN

System

- Pi Zero W + Pi Camera
- CrazyFlie (from BitCraze.io)
  - Optical flow sensor (X,Y position).
  - Laser-based ranging (Z position).



1. Mobile Real-time Video Streaming

- PHY/MAC: 802.11n
- NET: IP
- TRANS: PRRT

Goal: Reliable & Timely Video Stream

2. Edge-based Remote Control

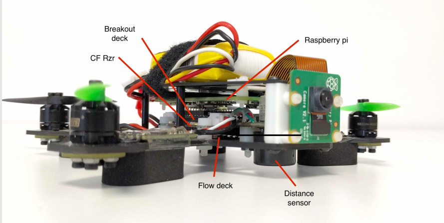
- PHY/MAC: CrazyRadio
- NET: None
- TRANS: PRRT

Goal: Stable Flight with Minimal Control inside the Drone

LARN

System

- Pi Zero W + Pi Camera
- CrazyFlie (from BitCraze.io)
  - Optical flow sensor (X,Y position).
  - Laser-based ranging (Z position).



1. Mobile Real-time Video Streaming

- PHY/MAC: 802.11n
- NET: IP
- TRANS: PRRT

Goal: Reliable & Timely Video Stream

2. Edge-based Remote Control

- PHY/MAC: CrazyRadio
- NET: None
- TRANS: PRRT

Goal: Stable Flight with Minimal Control inside the Drone

Currently negotiating/prototyping together with BitCraze.
Evaluation starting **approx. Jun/Jul'18**.

Accepted Publications

- Gil Pereira, Pablo; Schmidt, Andreas; Herfet, Thorsten: **"Cross-Layer Effects on Training Neural Algorithms for Video Streaming"**, 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), Amsterdam, Netherlands, June 2018
- Reif, Stefan; Schröder-Preikschat, Wolfgang: **"A Predictable Synchronisation Algorithm (Poster)"**, 23rd Annual Symposium on Principles and Practice of Parallel Programming (PPoPP), Vienna, Austria, February 2018

Accepted Publications

- Gil Pereira, Pablo; Schmidt, Andreas; Herfet, Thorsten: **"Cross-Layer Effects on Training Neural Algorithms for Video Streaming"**, 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), Amsterdam, Netherlands, June 2018
- Reif, Stefan; Schröder-Preikschat, Wolfgang: **"A Predictable Synchronisation Algorithm (Poster)"**, 23rd Annual Symposium on Principles and Practice of Parallel Programming (PPoPP), Vienna, Austria, February 2018

Publications Under Review

- Reif, Stefan; Schmidt, Andreas; Hönig, Timo; Herfet, Thorsten; Schröder-Preikschat, Wolfgang: **"Differential Energy-Efficiency and Timing Analysis for Real-Time Networks"**, 16th International Workshop on Real-Time Networks (ECRTS RTN), Barcelona, Spain, July 2018
- Schmidt, Andreas; Reif, Stefan; Gil Pereira, Pablo; Hönig, Timo; Herfet, Thorsten; Schröder-Preikschat, Wolfgang: **"Cross-Layer Pacing for Predictable Low-Latency Communication in Edge Computing"**, USENIX Workshop on Hot Topics in Edge Computing (HotEdge), Boston, USA, July 2018.

LARN

- ✔ **PRRT**[1] available for control and video applications as **open source**.
  - Straightforward usage for **C, Python, and Gstreamer** projects.
  - Compatibility with **Linux** on ARM and x86-64 platforms.
- ✔ PRRT packed within the **RNA**, running on different embedded platforms.
- ✔ Fine-grained **analysis** for causes of latency and jitter using **X-Lap**[2].
- ✔ Reduced **network layer latency and jitter** using...
  - hybrid error correction (FEQ + ARQ) by avoiding round-trips and
  - cross-layer pacing with congestion control to avoid queuing delays.
- ✔ Reduced **system layer latency and jitter** by...
  - efficient system architecture, and
  - wait-free synchronisation between threads.
- ✔ **Results** documented in **7 accepted publications** (2 conferences, 3 workshops, 2 posters) and **2 publications under submission**.
- ✔ Supervision of **3 successful master theses**.

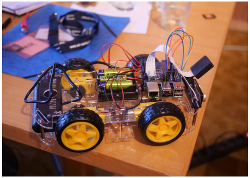
[1] http://prrt.larn.systems
[2] http://xlap.larn.systems

- ✔ **PRRT**[1] available for control and video applications as **open source**.
  - Straightforward usage for **C, Python, and Gstreamer** projects.
  - Compatibility with **Linux** on ARM and x86-64 platforms.
- ✔ PRRT packed within the **RNA**, running on different embedded platforms.
- ✔ Fine-grained **analysis** for causes of latency and jitter using **X-Lap**[2].
- ✔ Reduced **network layer latency and jitter** using...
  - hybrid error correction (FEQ + ARQ) by avoiding round-trips and
  - cross-layer pacing with congestion control to avoid queuing delays.
- ✔ Reduced **system layer latency and jitter** by...
  - efficient system architecture, and
  - wait-free synchronisation between threads.
- ✔ **Results** documented in **7 accepted publications** (2 conferences, 3 workshops, 2 posters) and **2 publications under submission**.
- ✔ Supervision of **3 successful master theses**.

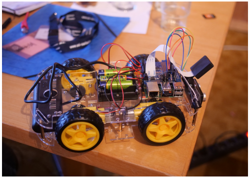
The LARN Team
2 PIs, 1 PostDoc, 3 PhD students and 3 student assistants

[1] http://prrt.larn.systems
[2] http://xlap.larn.systems

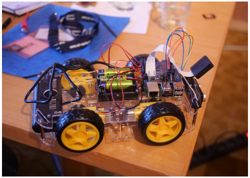
- RNA Applications and Scenarios
  - Autonomous driving car & . . .
  - . . . platooning
  - Drone for video streaming and control



- RNA Applications and Scenarios
  - Autonomous driving car & ...
  - ... platooning
  - Drone for video streaming and control

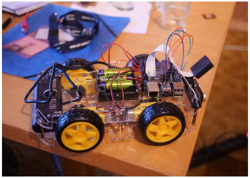


- X-Lap
  - Automated detection of causes for latency and jitter
  - Correlation between energy demand $\Longleftrightarrow$ processing speed



- RNA Applications and Scenarios
  - Autonomous driving car & ...
  - ...platooning
  - Drone for video streaming and control
- X-Lap
  - Automated detection of causes for latency and jitter
  - Correlation between energy demand $\Longleftrightarrow$ processing speed
- PRRT
  - Optimize error control for embedded platforms
  - Transparent transmission segmentation evaluations
  - Multicast support (NAKs, feedback suppression)

LARN



- RNA Applications and Scenarios
  - Autonomous driving car & ...
  - ... platooning
  - Drone for video streaming and control
- X-LAP
  - Automated detection of causes for latency and jitter
  - Correlation between energy demand $\Longleftrightarrow$ processing speed
- PRRT
  - Optimize error control for embedded platforms
  - Transparent transmission segmentation evaluations
  - Multicast support (NAKs, feedback suppression)
- Operating System Support
  - Time-predictable synchronisation
  - Cross-layer resource management

- Improve crosscutting system properties
  - focus on **energy efficiency**: impact of runtime adaptations
  - non-functional properties of networked systems (i.e. RNAs)
  - system configuration of individual RNAs (i.e. local scope)

    $\Updownarrow$

    energy demand/latency of overall system (i.e. global scope)

- Improve crosscutting system properties
  - focus on **energy efficiency**: impact of runtime adaptations
  - non-functional properties of networked systems (i.e. RNAs)
  - system configuration of individual RNAs (i.e. local scope)

    $\Updownarrow$

    energy demand/latency of overall system (i.e. global scope)

- Identification and proactive avoidance of bottlenecks within system stack
  - build „strain reliefs" to **avoid emergence of bottlenecks**
  - proactively exploit a priori knowledge (i.e. system design)
  - cooperative system-analysis (i.e. ahead of runtime + at runtime)

LARN

- Improve crosscutting system properties
  - focus on **energy efficiency**: impact of runtime adaptations
  - non-functional properties of networked systems (i.e. RNAs)
  - system configuration of individual RNAs (i.e. local scope)

    ⇕

    energy demand/latency of overall system (i.e. global scope)

- Identification and proactive avoidance of bottlenecks within system stack
  - build „strain reliefs" to **avoid emergence of bottlenecks**
  - proactively exploit a priori knowledge (i.e. system design)
  - cooperative system-analysis (i.e. ahead of runtime + at runtime)

- Explore adaptation of Phase 1 research results to related research areas
  - **edge-computing environments** (WIP): improve latency of edge components
  - consider power-demand constraints (i.e. low power IoT devices)
  - extended interweaving of network protocols and operating-system components

Conclusion

Network

- ✔ PRRT – Predictable protocol
- ✔ Network pacing

Operating System

- ✔ X-LAP – Timing measurement
- ✔ Wait-free synchronisation

+

Reliable Network Atom (RNA)
(Embedded Edge-Server, Car, Drone, ...)

LARN

Network

- ✔ PRRT – Predictable protocol
- ✔ Network pacing

Operating System

- ✔ X-LAP – Timing measurement
- ✔ Wait-free synchronisation

+

- Integration
- Optimisation
- Cross-layer pacing

Reliable Network Atom (RNA)

(Embedded Edge-Server, Car, Drone, ...)

LARN

Network

- ✔ PRRT – Predictable protocol
- ✔ Network pacing

Operating System

- ✔ X-LAP – Timing measurement
- ✔ Wait-free synchronisation

- Integration
- Optimisation
- Cross-layer pacing

+

Reliable Network Atom (RNA)
(Embedded Edge-Server, Car, Drone, ...)

Thank you for your attention. Questions?