LARN

Latency- and Resilience-Aware Networking

ΔELTA: Differential Energy-Efficiency, Latency, and Timing Analysis for Real-Time Networks

RTN 2018

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

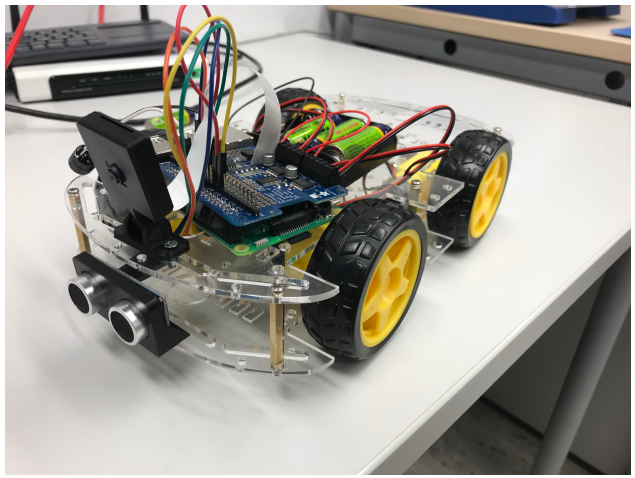
Andreas Schmidt, Thorsten Herfet
Telecommunications Lab
Saarland Informatics Campus - Saarbrücken

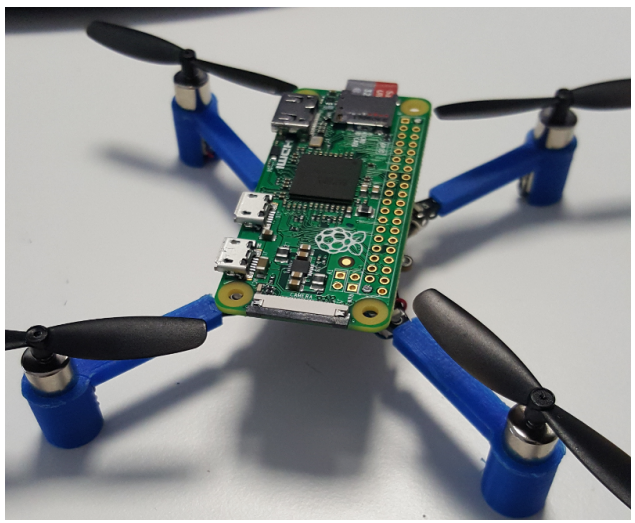
SIC Saarland Informatics Campus

DFG

FAU FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

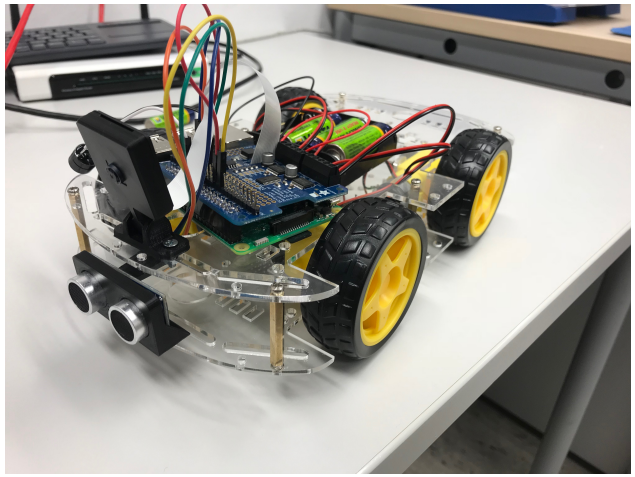
Motivation

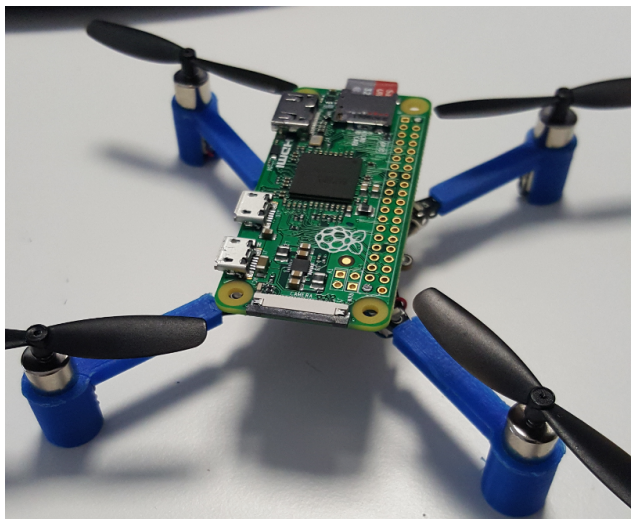




- Modern CPN/IoT systems are complex
  - Application → Image processing, ...
  - Hardware → Multi-core, caches, ...
  - Network → Internet, WiFi, ...
- Energy supply is limited
- Static timing and energy analysis techniques reach their limits
  - Some aspects are unknown at compile time
  - Random environmental influences

Motivation



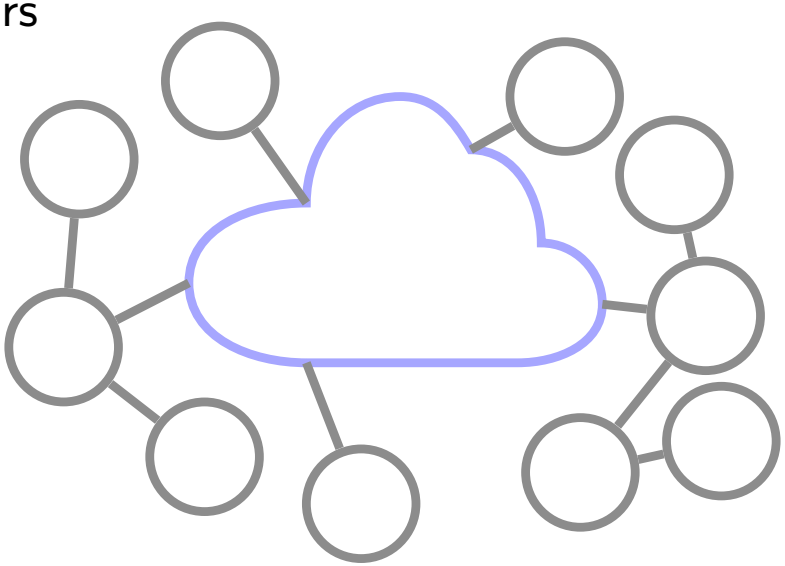


- Modern CPN/IoT systems are complex
  - Application → Image processing, ...
  - Hardware → Multi-core, caches, ...
  - Network → Internet, WiFi, ...
- Energy supply is limited
- Static timing and energy analysis techniques reach their limits
  - Some aspects are unknown at compile time
  - Random environmental influences

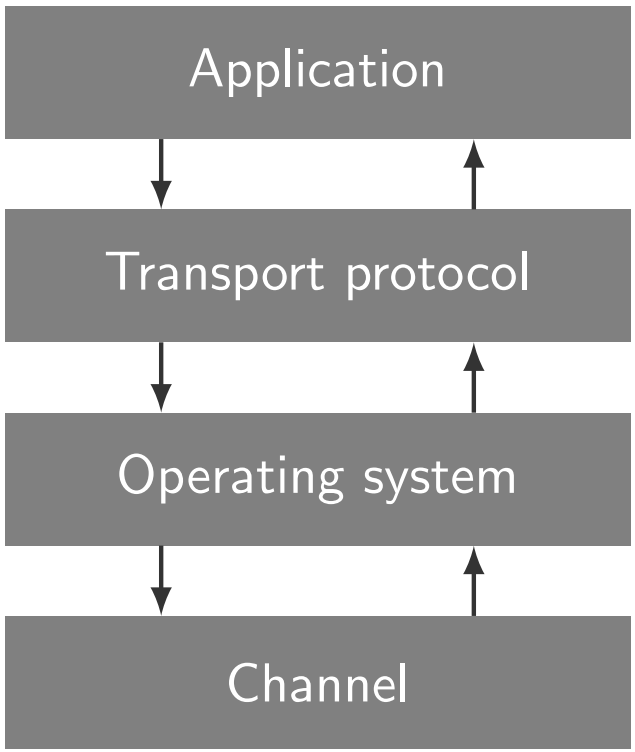
Need for on-line analysis

PRRT:

- Soft real-time network protocol
- Latency awareness
- Congestion control
- Adaptive hybrid error control
- Bindings for C, Python, and others
- Source code is available:
  → `http://prrt.larn.systems`



LARN

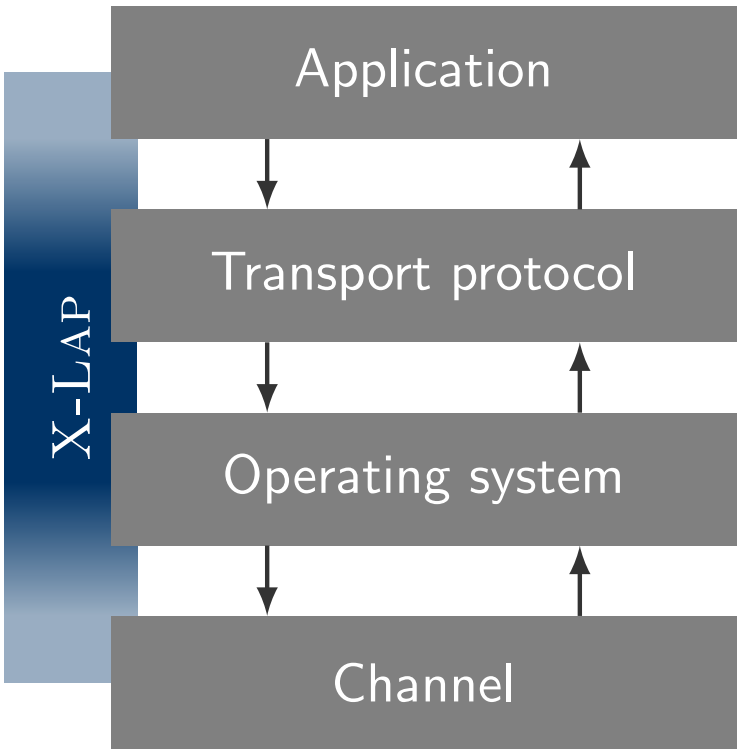


Time measurement:

1. Insert timestamping points into real-time network system
2. Execute experiment
3. Collect and combine traces
4. Visualize traces

Timing information:

- Per-packet
- Cross-layer
- Multi-node

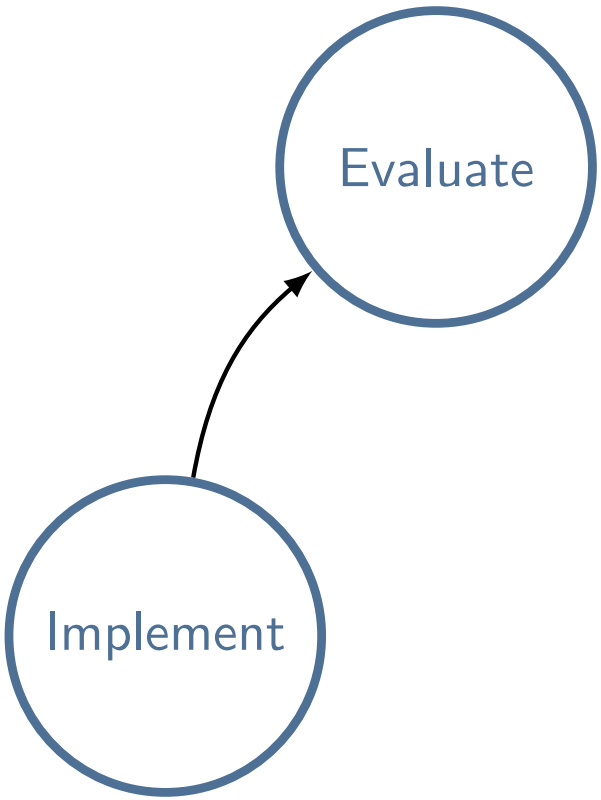


Time measurement:

1. Insert timestamping points into real-time network system
2. Execute experiment
3. Collect and combine traces
4. Visualize traces

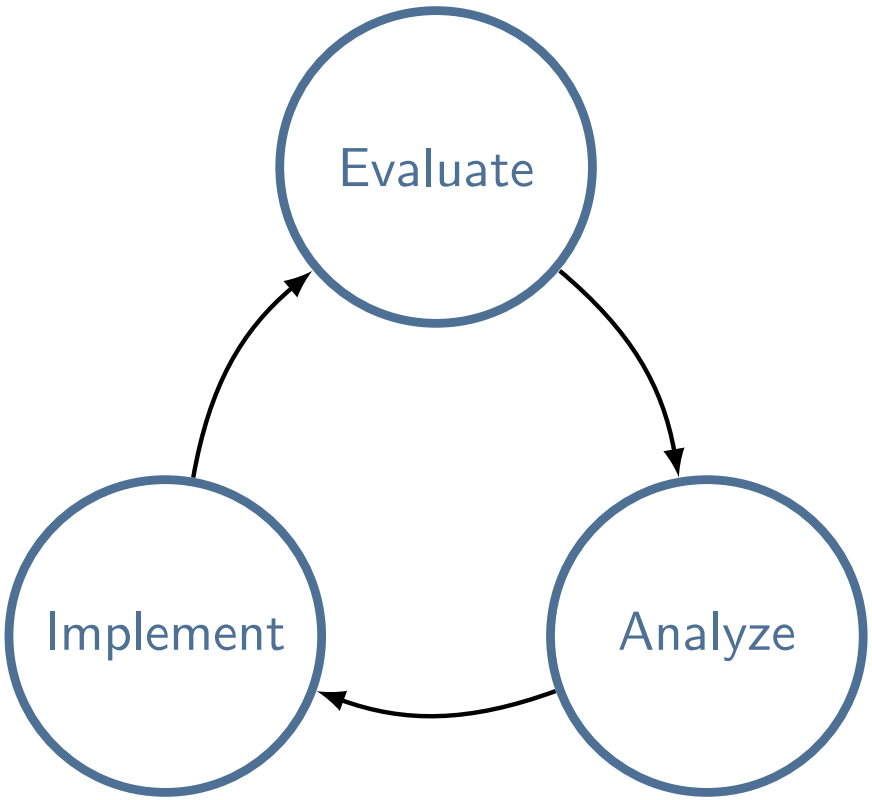
Timing information:

- Per-packet
- Cross-layer
- Multi-node



Approach:

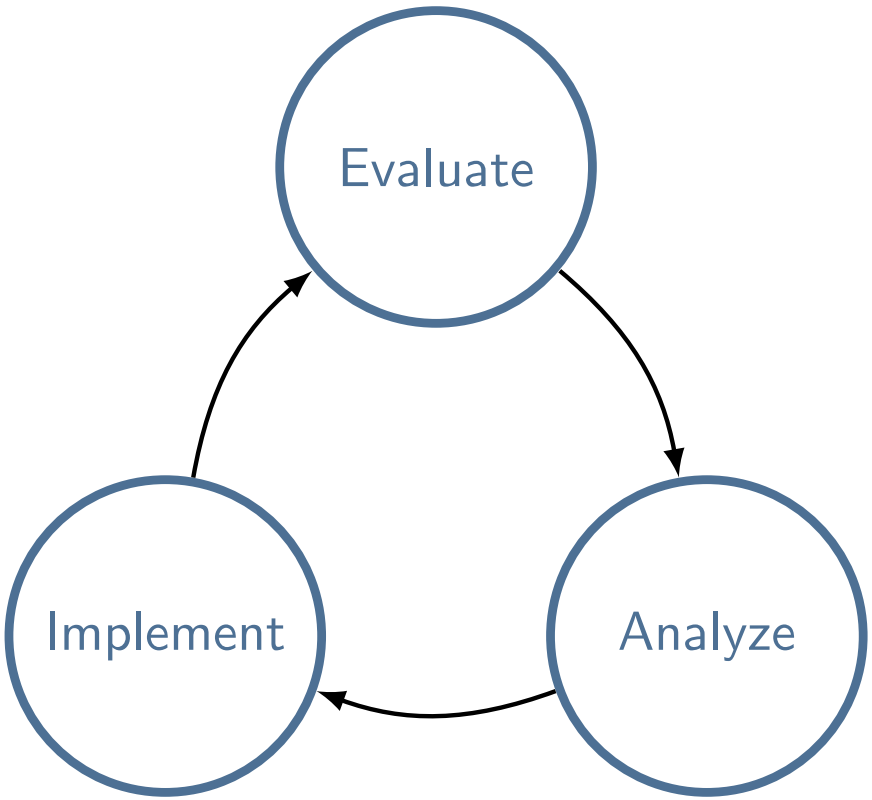
- Measurement-based protocol evaluation



Approach:

- Measurement-based protocol evaluation
- Derive meaningful information
- Support protocol design

Approach



Approach:

- Measurement-based protocol evaluation
- Derive meaningful information
- Support protocol design

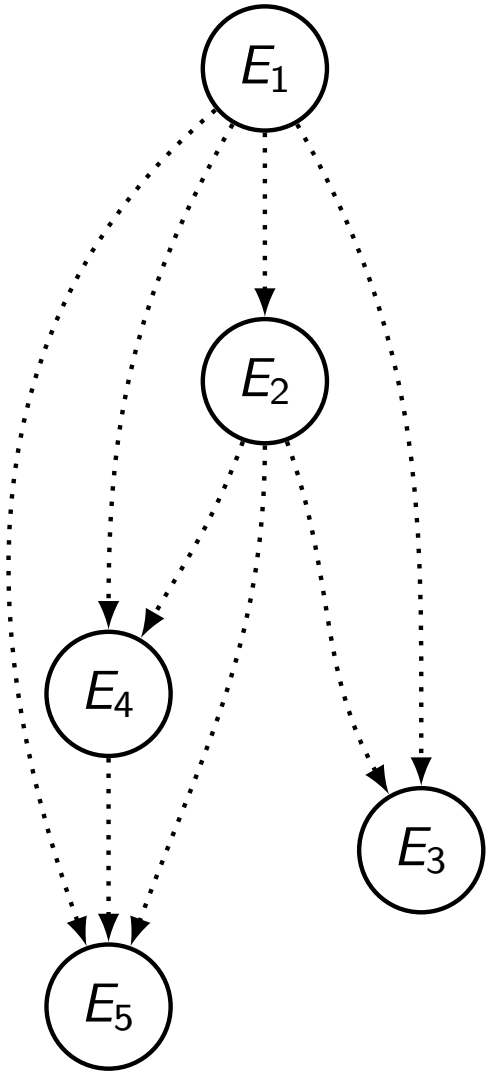
Challenges:

- ☐ Identify *critical code parts*
  - → Latency, Jitter, Energy
- ☐ Analyze code modifications
- ☐ Automate feedback

Outline

Outline

Control-Flow Reconstruction



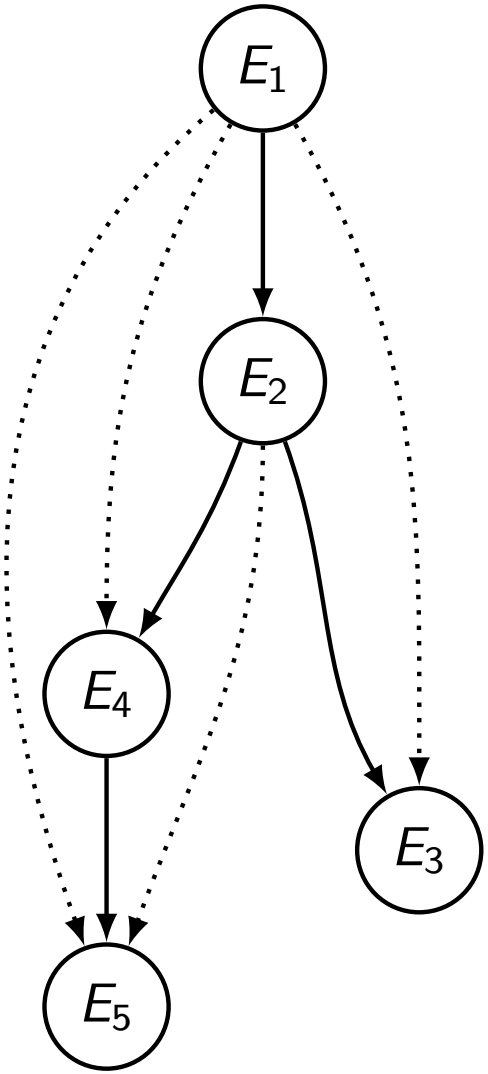
Goal:

- Reconstruct protocol information from timestamps

Control flow reconstruction:

1. Compute *happens-before* relation of events

Control-Flow Reconstruction



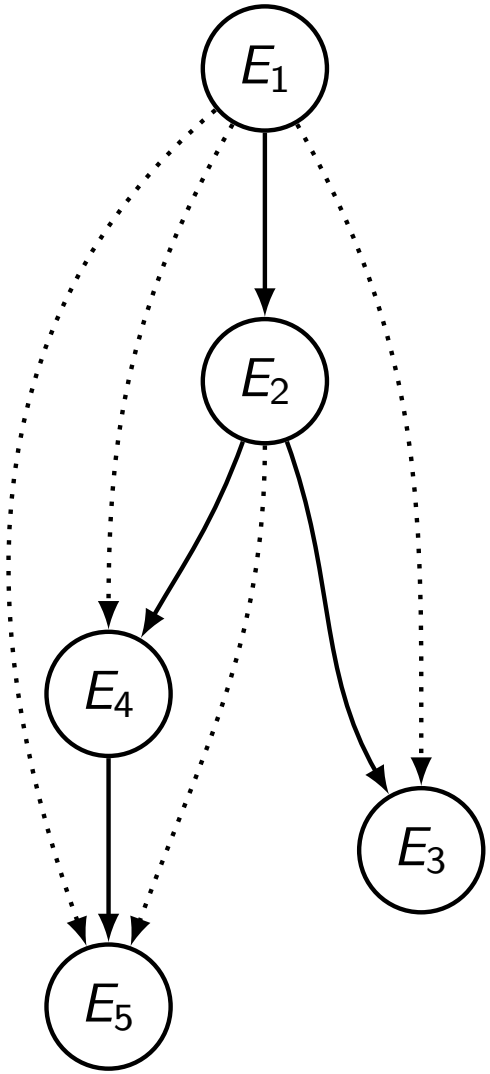
Goal:

- Reconstruct protocol information from timestamps

Control flow reconstruction:

1. Compute *happens-before* relation of events
2. Compute *happens-directly-before* relation of events

Control-Flow Reconstruction



Goal:

- Reconstruct protocol information from timestamps

Control flow reconstruction:

1. Compute *happens-before* relation of events
2. Compute *happens-directly-before* relation of events

Edges represent:

- Code segments $\langle E_i, E_j \rangle$
- Thread communication
- False-positives

LARN

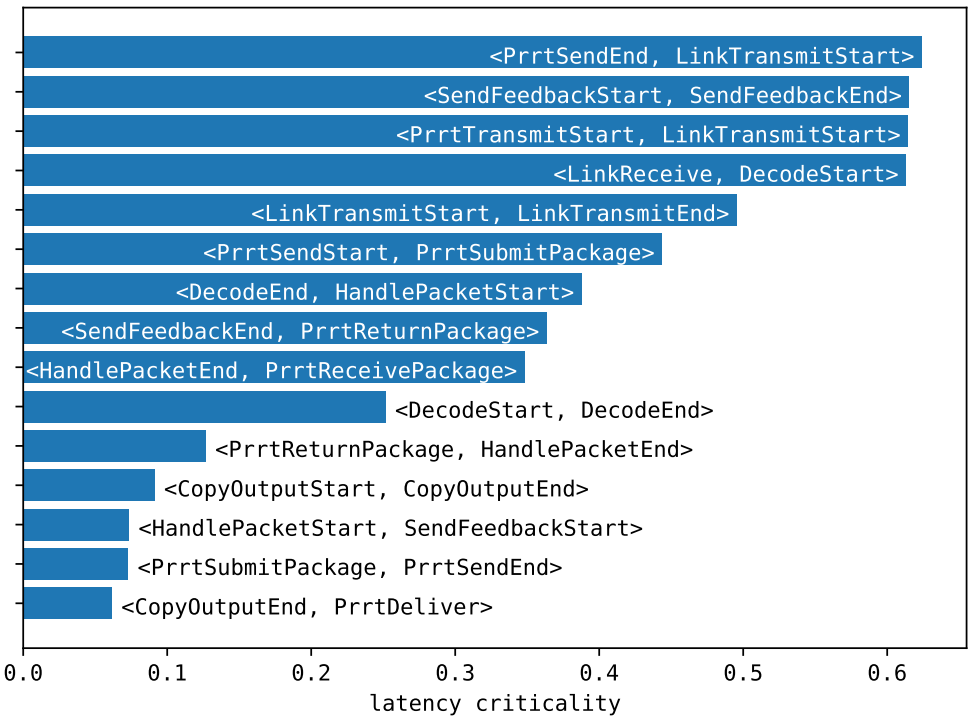
Goals:

- Filter out false-positive segments
- Identify code segments worth optimizing

Detect *relevant* code segments:

1. Compute the correlation between the segment latency and the E2E latency
2. Ignore segments with negative correlation

Latency Criticality Analysis



Goals:

- Filter out false-positive segments
- Identify code segments worth optimizing

Detect *relevant* code segments:

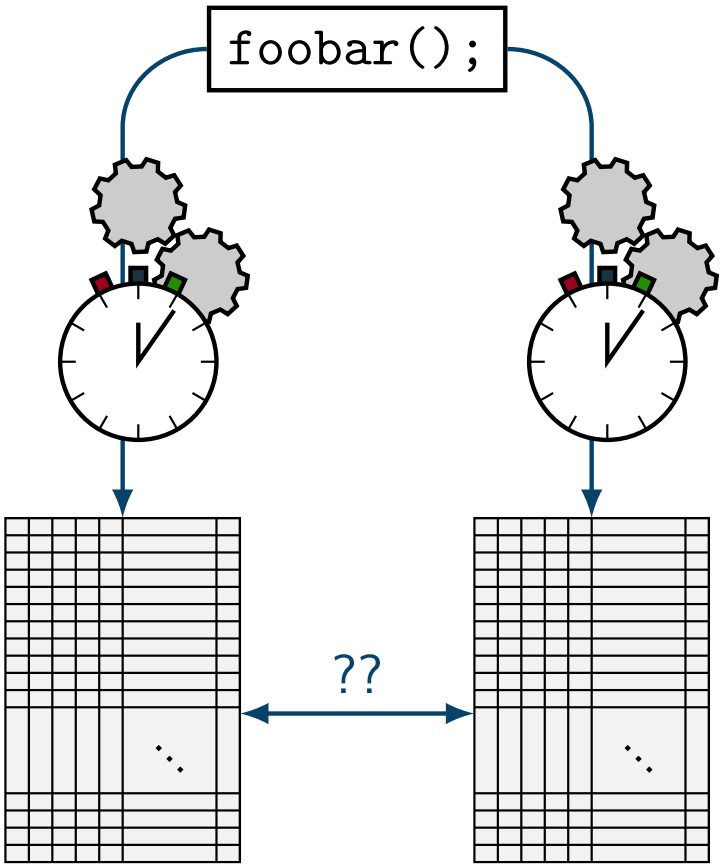
1. Compute the correlation between the segment latency and the E2E latency
2. Ignore segments with negative correlation

LARN

Goals:

- Identify code segments with unpredictable timing
- “What happens if ...” analysis

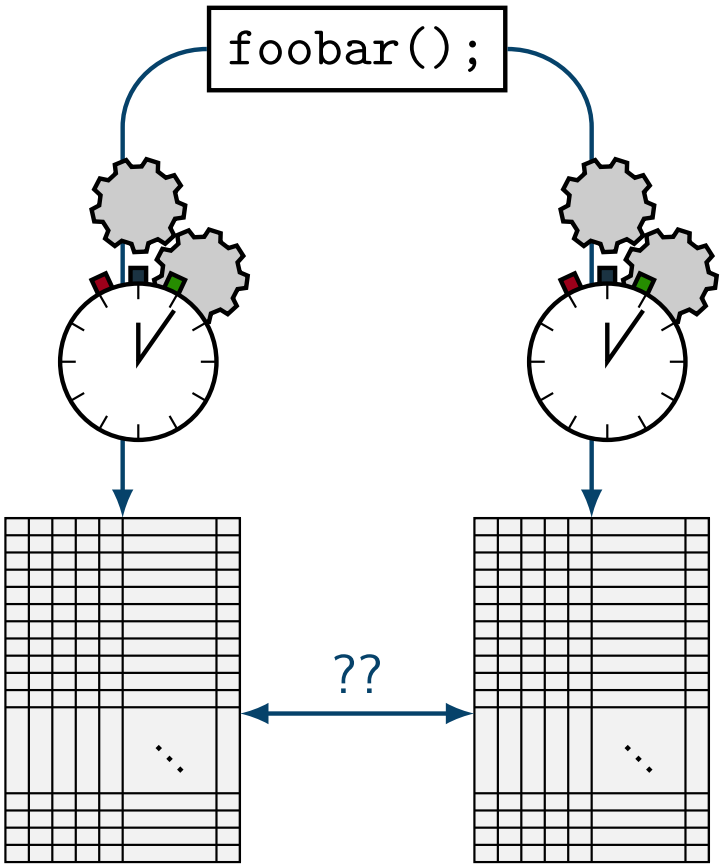
Timing Reproducibility Analysis



Goals:

- Identify code segments with unpredictable timing
- “What happens if ...” analysis

Timing Reproducibility Analysis



Goals:

- Identify code segments with unpredictable timing
- "What happens if ..." analysis

Timing reproducibility analysis:

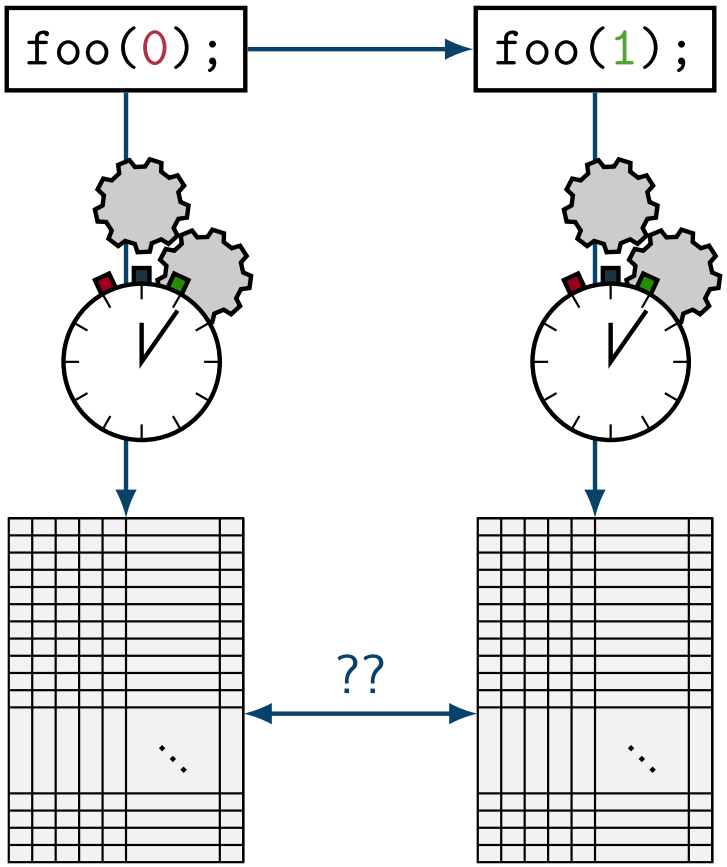
1. Evaluate the code twice
2. Apply the *k-sample Anderson-Darling-Test* on the traces
3. For each code segment, test decides: "similar" or "different"

LARN

Goals:

- Evaluate the influence of code changes
- Interferences are complex

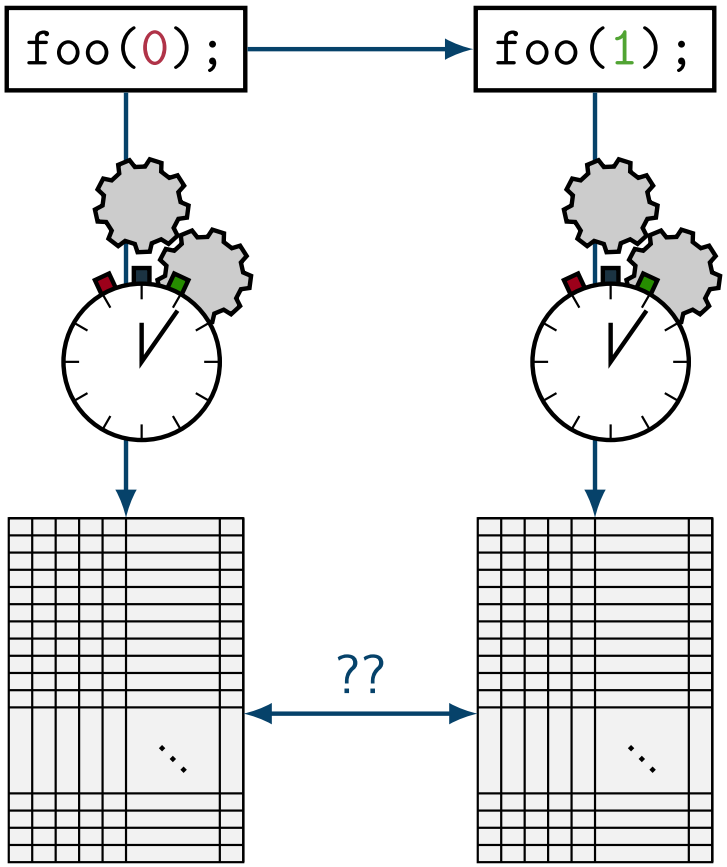
Code Modification Analysis



Goals:

- Evaluate the influence of code changes
- Interferences are complex

Code Modification Analysis



Goals:

- Evaluate the influence of code changes
- Interferences are complex

Modification impact analysis:

1. Evaluate the original and the modified versions of the protocol
2. Apply the *k-sample Anderson-Darling-Test* on the traces
3. For each code segment, test decides: “similar” or “different”

LARN

```
<PrrtSendPacketStart,PrrtSendPacketEnd> has not changed
<CopyOutputStart,CopyOutputEnd>         has not changed
<SendFeedbackStart,SendFeedbackEnd>     has not changed
...

<PrrtSubmitPackage,PrrtSendEnd> has changed:
  2.85±0.91 µs → 4.94±0.58 µs

<LinkReceive,DecodeStart> has changed:
  2.36±3.80 µs → 3.56±4.88 µs
...
```

Outline

Energy Efficiency Analysis

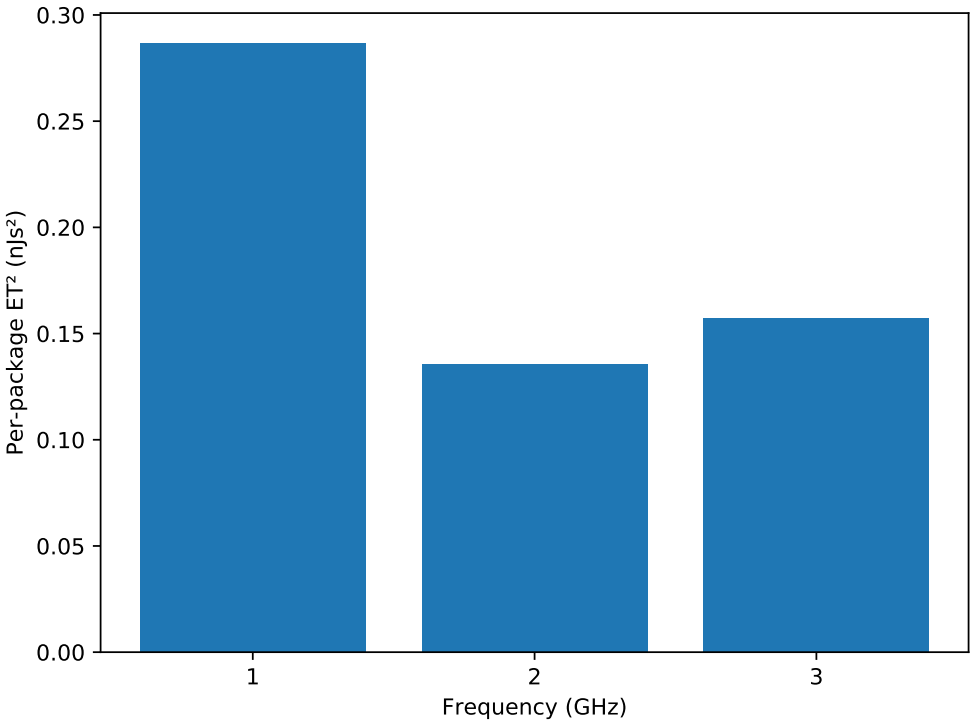
Goals:

- Identify energy-efficient hardware configurations

Energy efficiency evaluation:

1. Evaluate *slow* and *fast* configurations
   - $\rightarrow$ Measure execution times and energy demand
2. Compute ET^2 metric per packet

Energy Efficiency Analysis



Goals:

- Identify energy-efficient hardware configurations

Energy efficiency evaluation:

1. Evaluate *slow* and *fast* configurations
   → Measure execution times and energy demand
2. Compute ET^2 metric per packet

LARN

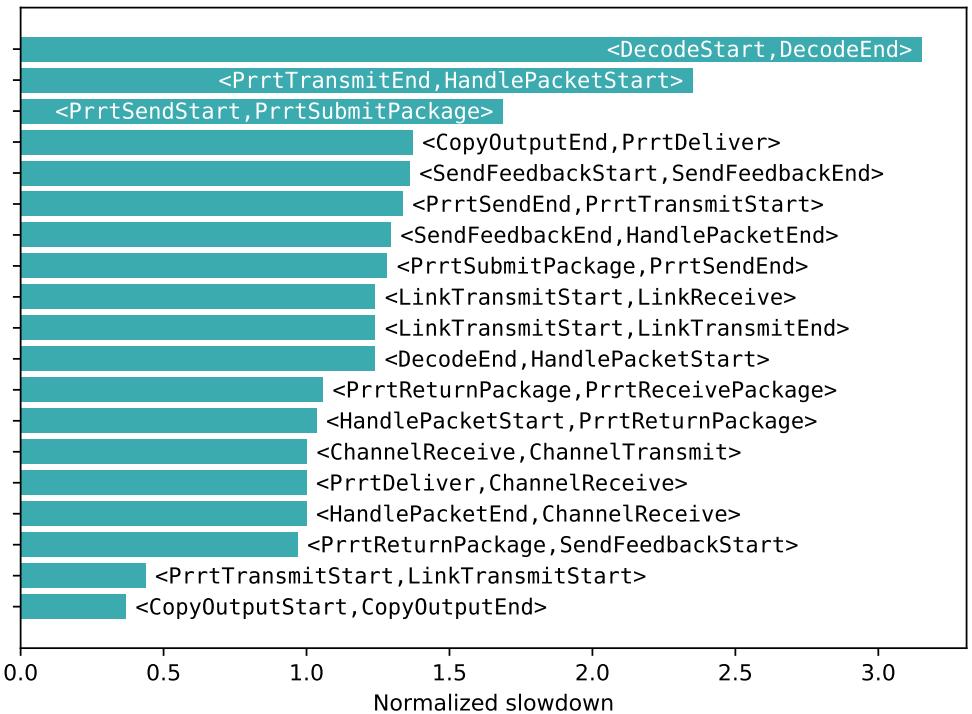
Goals:

- Identify code segments susceptible to slowdown

Slowdown computation:

1. Evaluate *slow* and *fast* configurations
2. Compute $S(\langle E_i, E_j \rangle) = \frac{D_{slow}(\langle E_i, E_j \rangle)}{D_{fast}(\langle E_i, E_j \rangle)}$
3. Normalize with $S(\text{End-to-End})$

Slowdown Analysis



Goals:

- Identify code segments susceptible to slowdown

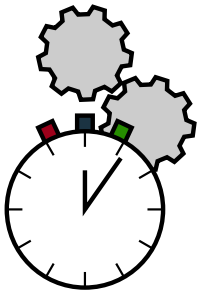
Slowdown computation:

1. Evaluate *slow* and *fast* configurations
2. Compute $S(\langle E_i, E_j \rangle) = \frac{D_{slow}(\langle E_i, E_j \rangle)}{D_{fast}(\langle E_i, E_j \rangle)}$
3. Normalize with $S(\text{End-to-End})$

Outline

Summary

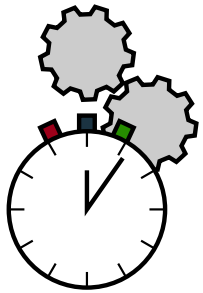
ΔELTA



- Analyze the run-time behavior of real-time networks
- Provide valuable feedback to improve implementations
- Based on X-LAP

Summary

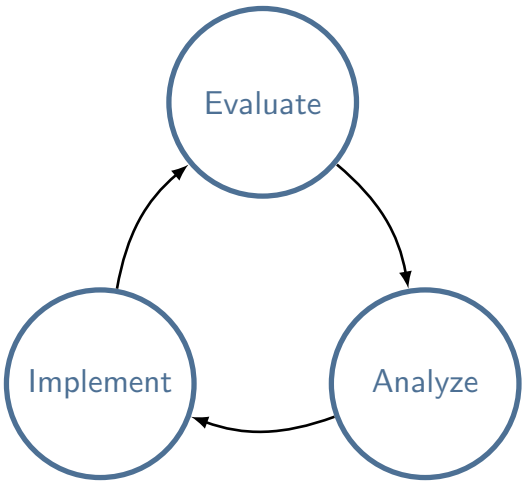
ΔELTA



- Analyze the run-time behavior of real-time networks
- Provide valuable feedback to improve implementations
- Based on X-LAP

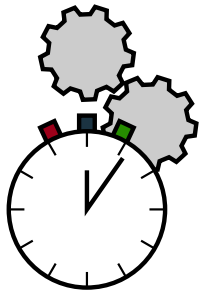
Analyse energy, latency, and jitter

- ☑ Identify *problematic* code parts
- ☑ Analyze code modifications
- ☑ Automate feedback



Summary

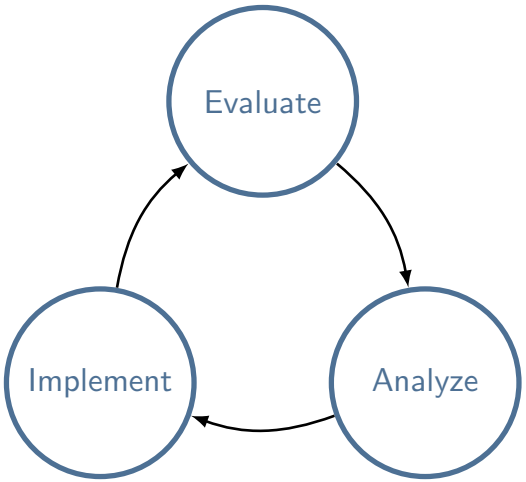
ΔELTA



- Analyze the run-time behavior of real-time networks
- Provide valuable feedback to improve implementations
- Based on X-LAP

Analyse energy, latency, and jitter

- ☑ Identify *problematic* code parts
- ☑ Analyze code modifications
- ☑ Automate feedback

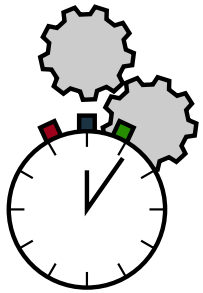


Source code available online:
→ `http://xlap.larn.systems`

Summary

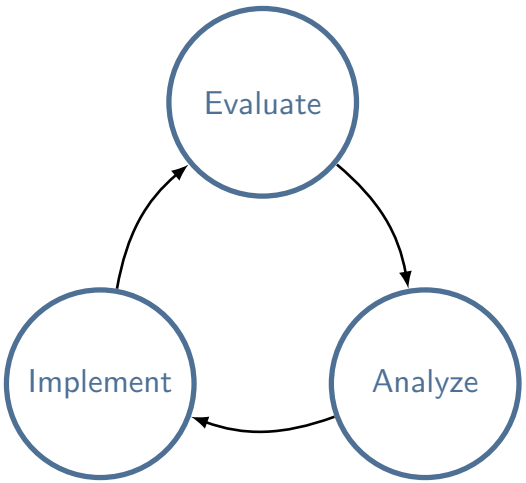
ΔELTA

- Analyze the run-time behavior of real-time networks
- Provide valuable feedback to improve implementations
- Based on X-LAP



Analyse energy, latency, and jitter

- ☑ Identify *problematic* code parts
- ☑ Analyze code modifications
- ☑ Automate feedback



Source code available online:
→ `http://xlap.larn.systems`

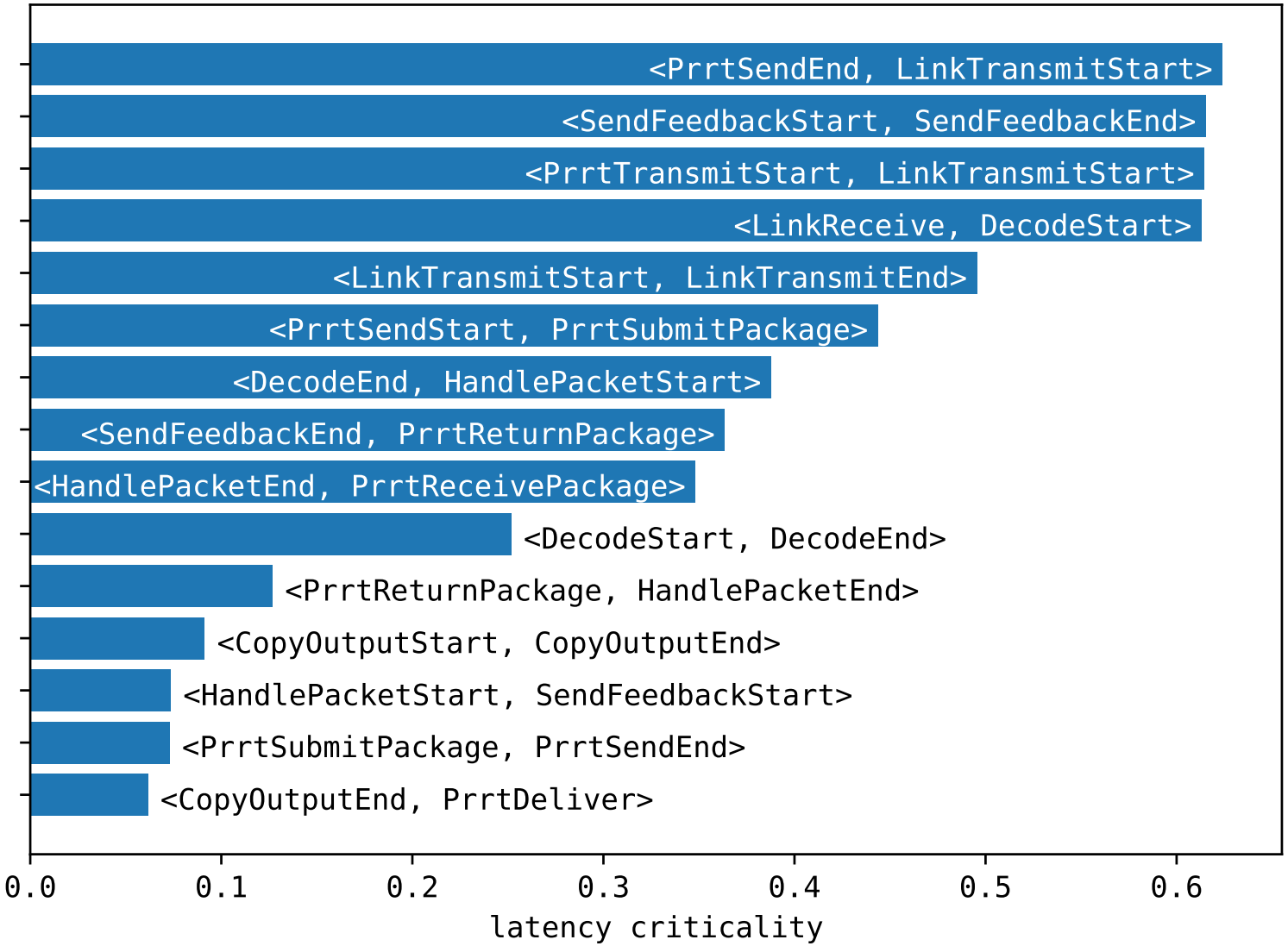
Thank you for your attention. Questions?

Acknowledgements

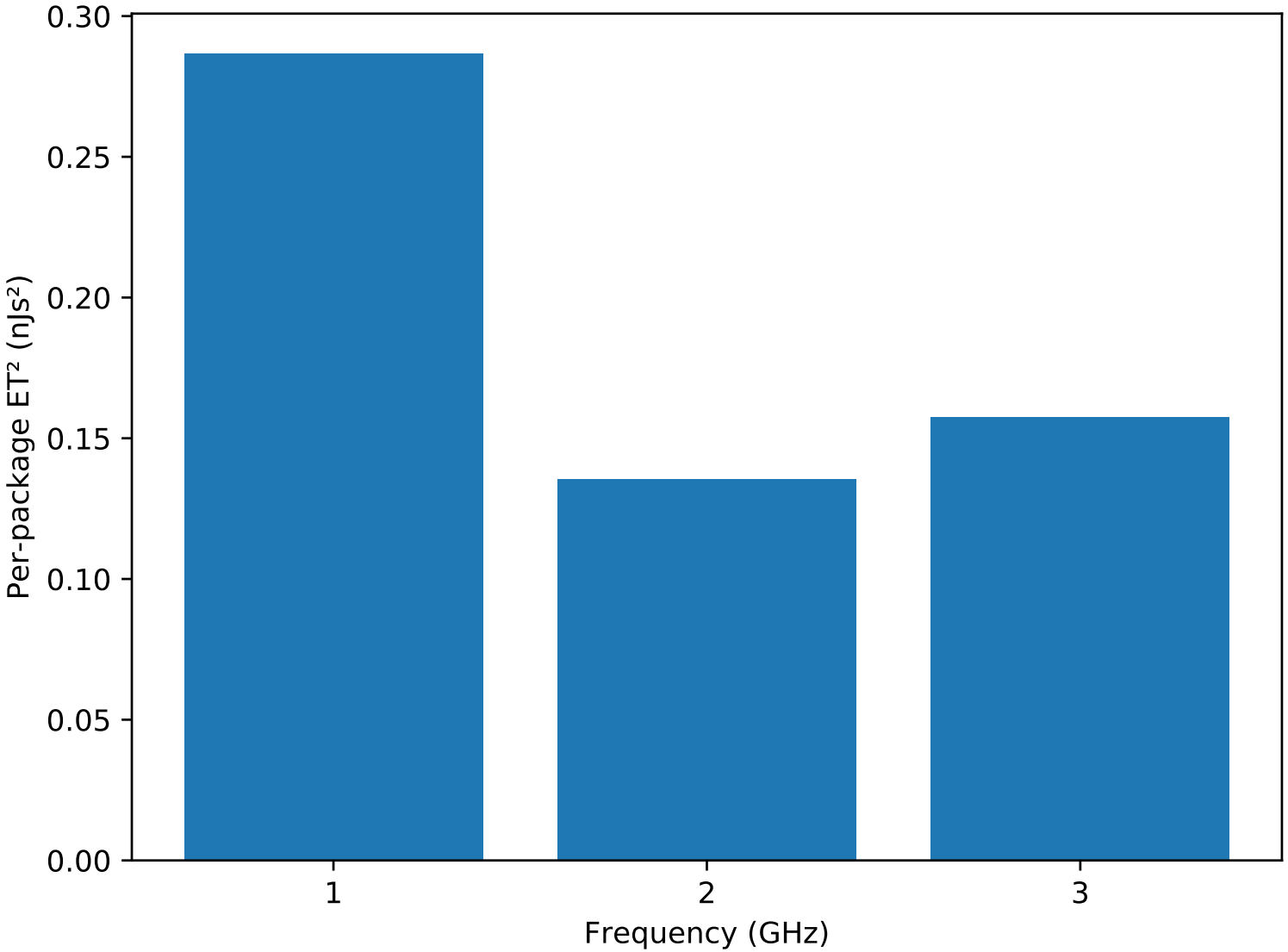
Supported by:

HE 2584/4-1
SCHR 603/15-1

Latency Criticality Analysis



Energy Efficiency Analysis



Slowdown Analysis