Latency- and Resilience-aware Transport for the Industrial Internet
KuVS Summer School Industrial Internet 2018

Andreas Schmidt
Telecommunications Lab
Saarland Informatics Campus - Saarbrücken
Motivation

Internet (i.e. TCP/IP) is...

best-effort := no time-awareness but fault-tolerant.

Industrial Internet requires...

- ... fault-tolerance to cope with local failures.
- ... time-awareness to dependably interact with physical processes.

Industrial Internet protocols MUST provide time-awareness and reliability.
Research Background

DFG SPP1914 “Cyber-Physical Networking”

- Joint research on systems, network, and control engineering.

“Latency- And Resilience-aware Networking” (Sep. 2016 - now)

- Telecommunications Chair at Saarland Informatics Campus
- Department of Computer Science 4 (Distributed Systems and Operating Systems) Friedrich-Alexander-Universität Erlangen-Nürnberg
Problem Definition

State-of-the-Art ($\varepsilon$: Error, $T$: Latency)

- **(Simplified) Shannon:** $(\varepsilon \to 0) \Rightarrow (T = \infty)$
- **UDP** can neither make statements about $\varepsilon$ nor bound $T$.
- **TCP** (and others with $\varepsilon = 0$) cannot bound $T$.

Industrial Internet & Multimedia Applications require bounds

- **Latency:** $\exists T_{max}: \forall T_i: T_i < T_{max} \wedge T_{max} \neq \infty$
- **Resilience:** $\varepsilon \approx 0$, but $\varepsilon \neq 0$

**Goal:** Develop an adaptive transport protocol that considers application requirements (latency and resilience bounds) and fulfills them if possible.
- **Reliable process-to-process real-time communication.**
  → Transport layer protocol.

- **Minimal requirements to lower layer.**
  → Rely on lower layer for forwarding, routing, process-multiplexing.
  → Allows to run on Ethernet / industrial busses.

- **No network knowledge required.**
  → Agnostic E2E mode.
  → Broad applicability, even on networks / links not under our control.

- **No network guarantees on channel parameters required.**
  → They are helpful, nevertheless.

- **Application must handle if constraints cannot be fulfilled currently.**
  → Channel is dynamic, so fulfilling constraints might be infeasible.
PRRT: Approaches

Measurement

- **Latency**: RTP-like
- **Loss**: Windowed packet tracking, Gilbert-Elliot model fitting
- **Data Rate**: Delivery Rate Estimation (IETF-Draft Cheng 2017)

Control

- **Error Control**: Hybrid ARQ / Hybrid Error Coding.
- **Congestion Control**: BBR-inspired.
- **Rate Control**: Packet pacing.
## PRRT: Availability

### Code
- Free Open Source Software
  - [http://prrt.larn.systems](http://prrt.larn.systems)
- C-Code + API (library)
- Python Bindings
- Gstreamer-Plugin

### Platform Support
- **Requirement:** glibc, pthreads
- Raspberry Pi (3B, Zero W): Raspbian
- Odroid XU4: Ubuntu (16.04, 18.04)
Timing and predictability are crucial for the Industrial Internet.

Background

- Validation requires per-packet timing information.
- Run-time evaluation necessary, because ...
  - ... hardware details are not known in advance.
  - ... this information could be used to adapt protocol operation at runtime.

Idea: Instrument code with timestamping calls that have minimal impact on the results.

Solution: X-Lap
Cross-Layer Analysis Tool
(presented at Int. Workshop on Real-Time Networks (RTN17 & 18) at ECRTS)
X-Lap: Architecture

**Capturing**
- Timestamping functions
- ⇒ Time- & cyclestamps
- C code

**Analysis**
- Data analysis
- ⇒ Latency and jitter
- Python code

- Trace every packet
- Minimize run-time interference
- Embedded into protocol source code

- Single-packet traces
- Jitter amongst packets
- Latency criticality
- Correlation analysis
Summary

PRRT
- Predictably reliable real-time transport protocol.
- Error, congestion and rate control.
- Available for C, Python & Gstreamer.

X-Lap
- Validation tool.
- Packet-level tracing & post-processing.
- Timing & resilience analysis.

For more information: [http://larn.systems](http://larn.systems)

Thank you for your attention. Questions?
Backup
General Approach (adapted re-implementation of Google’s BBR)

- Measure...
  - $T_{ch}$ (channel propagation delay) and $R_{ch}$ (channel bottleneck data rate).
  - Keep $cwnd$ at one BDP ($T_{ch} \cdot R_{ch}$).
  - Keep $pace$ between packets of size $L$ by waiting for $\frac{L}{R_{ch}}$.

Adaptiveness

- Control sending $pace$ and $cwnd$ to do alternating measures of both channel parameters.
- Handle spurious loss events (caused by physical loss) differently to burst.

Approach not yet “perfect”, but controlling $cwnd$ and $pace$ is optimal strategy.
Hybrid Error Coding

- Configuration: $C = (n = 7, k = 4, \bar{n} = [1, 2])$

Adaptive Hybrid Error Coding

- Consider application constraints: $\varepsilon_{max}, T_{max}$.
- Measure channel parameters: $\varepsilon_{ch}, T_{ch}, R_{ch}$.
- Search for $C_{optimal}$ which...
  - ... fulfills application bounds.
  - ... minimizes caused redundancy (simplified: $\frac{n-k}{k}$).
X-Lap: Packet Traces
X-Lap: Correlation

![Graphs showing correlation between various timings and metrics.](image-url)
X-Lap: Latency Criticality

- <PrrtSendEnd, LinkTransmitStart>
- <SendFeedbackStart, SendFeedbackEnd>
- <PrrtTransmitStart, LinkTransmitStart>
- <LinkReceive, DecodeStart>
- <LinkTransmitStart, LinkTransmitEnd>
- <PrrtSendStart, PrrtSubmitPackage>
- <DecodeEnd, HandlePacketStart>
- <SendFeedbackEnd, PrrtReturnPackage>
- <HandlePacketEnd, PrrtReceivePackage>
- <DecodeStart, DecodeEnd>
- <PrrtReturnPackage, HandlePacketEnd>
- <CopyOutputStart, CopyOutputEnd>
- <HandlePacketStart, SendFeedbackStart>
- <PrrtSubmitPackage, PrrtSendEnd>
- <CopyOutputEnd, PrrtDeliver>
X-Lap: Slow-Down

Normalized slowdown

0.0 0.5 1.0 1.5 2.0 2.5 3.0

<PrrtTransmitEnd,HandlePacketStart>
<PrrtSendStart,PrrtSubmitPackage>
<PrrtSendStart,PrrtSubmitPackage>
<PrrtSendStart,PrrtSubmitPackage>
<PrrtSendEnd,PrrtTransmitStart>
<PrrtSubmitPackage,PrrtSendEnd>
<LinkTransmitStart,LinkReceive>
<LinkTransmitStart,LinkTransmitEnd>
DecodeEnd,HandlePacketStart>
<PrrtReturnPackage,PrrtReceivePackage>
(HandlePacketStart,PrrtReturnPackage>
(PrpPacketStart,PrpReturnPackage>
<ChannelReceive,ChannelTransmit>
<PrrtDeliver,ChannelReceive>
<HandlePacketEnd,ChannelReceive>
<PrrtReturnPackage,SendFeedbackStart>
<PrrtTransmitStart,LinkTransmitStart>
<CopyOutputStart,CopyOutputEnd>