TriCePS: Self-optimizing Communication for Cyber-Physical Systems

Jia L. Du[1], Stefan Linecker[1], Peter Dorfinger[1] and Reinhard Mayr[2]
[1]*Advanced Networking Center, Salzburg Research, Jakob Haringer Str. 5/3, Salzburg, Austria*
[2]*COPA-DATA GmbH, Karolingerstrasse 7B, Salzburg, Austria*

Keywords: Cyber-Physical Systems, Internet of Things, Interoperability, Self-adaptation, Protocol Negotiation, Protocol Optimization.

Abstract: The progress in energy-efficient, cost-effective and highly capable sensor-actuator electronics and data transmission technologies has been triggering a new phase in the digital transformation with potentially billions of cyber-physical systems connected in the Internet of Things in future. To fully harvest the potential of this development, a strategy for efficient, robust, interoperable and future-proof communication between a myriad of different CPS in a global network is essential. Such a strategy will have to cope with the desire for communication between potentially up front unknown systems using up front unknown communication networks under unknown conditions. Within the TriCePS project a framework and missing building blocks for adaptive communication for cyber-physical systems are designed and developed. The three main pillars will be Application Adaptation, Protocol Negotiation and Protocol Parameter Optimization. In this paper, the general concept and architecture as well as first results are presented.

1 INTRODUCTION

The term cyber-physical system (CPS) emerged around 2006, when it was coined by Helen Gill at the National Science Foundation in the United States (Lee, 2015). A cyber-physical system is a system that integrates computation with physical processes. Embedded, networked computers use sensors and actuators to interact with the physical world. Physical processes then affect computation and vice versa. In contrast to traditional embedded systems, a CPS is a network of interacting appliances instead of a standalone device (Minerva et al., 2015).

The CPS concept is strongly connected to the notion of the Internet of Things (IoT). IoT is a global infrastructure enabling advanced services and consisting of networked everyday objects which are connected through interoperable information and communication technologies (Xia et al.,). As IoT is about the interconnection of (everyday) objects (or things), it can be seen as related to the idea of ubiquitous computing that Mark Weiser came up with in 1988 (Weiser, 1999). IoT applications include smart home, wearables, connected car, smart grid, and many more.

The concepts of CPS and IoT have become possible due to the progress in highly capable, often embedded, cost effective, energy-efficient sensor-actuator and computation electronics and data transmission technologies.

1.1 Communication for CPS

With a myriad of network-connected entities, there will also be an immense number of different communication connections with varying qualities of connectivity. Caused by the stochastic behaviour of the underlying wired or wireless (usually IP) networks and applications that compete for bandwidth, the experienced quality of service (QoS) can be volatile. The size of the experienced variations depends on the ratio between the application requirements and the granted network resources. Managing this interdependence successfully, is a main challenge in comparison to existing systems, e.g. in the energy domain, which rely on exclusive and over-provisioned communication infrastructures. In addition to different qualities of connectivity, a multitude of used communication standards and a wide variety of hardware capabilities will also present a challenge.

A generally suitable CPS end system architecture will require new approaches, which can fulfill the adaptivity and openness needs for the integration of highly different systems under varying conditions.

Du, J., Linecker, S., Dorfinger, P. and Mayr, R.
TriCePS: Self-optimizing Communication for Cyber-Physical Systems.
DOI: 10.5220/0007696402410247
In *Proceedings of the 4th International Conference on Internet of Things, Big Data and Security (IoTBDS 2019)*, pages 241-247
ISBN: 978-989-758-369-8

1.2 Project Overview

The TriCePS project addresses the communication-related challenges for future cyber-physical systems. It aims for interoperable and efficient communication in such systems and uses adaptation and optimization as its basic strategies. The missing building blocks for adaptation and self-optimization that TriCePS aims to contribute can be described in form of three components (Figure 1):

- The first TriCePS component, “Application Adaption”, will inform the application layer about QoS values. The application then has the chance to adapt to the current conditions, e.g. by activating local pre-processing.
- The second TriCePS component, “Protocol Negotiation”, enables interoperability and efficient communication between end systems by allowing negotiating and choosing a communication protocol supported by all partners or even by downloading a protocol from a central repository.
- Finally, the third TriCePS component, “Protocol Parameter Optimization”, optimizes the parameters of the protocols in use for the current application, content and infrastructure.

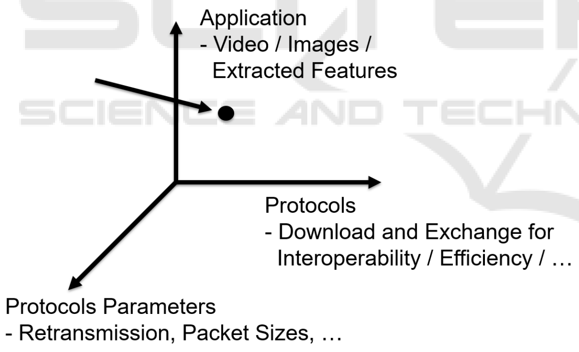
Application
- Video / Images / Extracted Features
Protocols
- Download and Exchange for Interoperability / Efficiency / …
Protocols Parameters
- Retransmission, Packet Sizes, …

Figure 1: Illustration of TriCePS ”Optimization Space”.

Such a comprehensive approach with a combination of these three components will provide future interconnected CPS with the means to establish interoperability and to efficiently and robustly communicate with each other in the future Internet of Things.

2 REQUIREMENTS AND GOALS

In this chapter the assumptions, requirements and adaptation mechanisms for TriCePS are presented which have served as basis for the architectural decisions in the next chapter.

2.1 Network Model, Constraints and Requirements

TriCePS assume a very general network model (Figure 2). When thinking about general, adaptive and future-proof CPS communication systems, we cannot assume in advance too much about either the nodes or the edges and have to be content with this very basic, abstract model. Potential communication nodes in the network include all kinds of devices such as servers, mobile devices, embedded devices. A given network route may cross multiple underlying logical and physical local or wide-area, wired or wireless networks. Every route can be thought of as having a variety of properties including current available bandwidth, round-trip time and number of hops. Some of these properties might be mostly static (like the number of hops between two devices in a wired, local network), whilst others may change continuously (like the available bandwidth between two devices when other network flows are competing for resources).

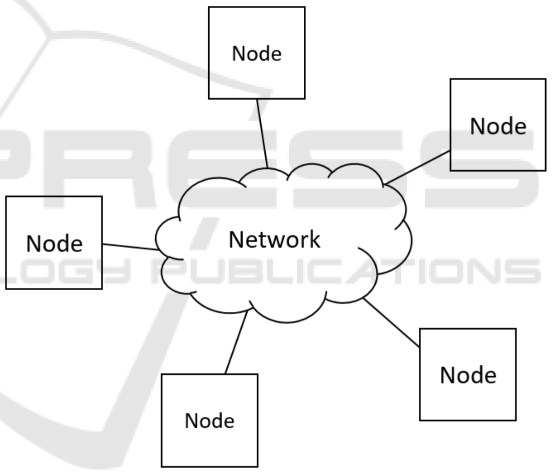

Figure 2: TriCePS network model.

Additionally, some further general assumptions and requirements are:

- There is no control over the network components, features or settings between two nodes.
- The route between two nodes can be used by potentially many other devices, components and services too.
- There is no general possibility to control or modify node network stack behaviour.
- There is the possibility to influence user space application behaviour on nodes.
- A focus will be put on TCP traffic.

2.2 Adaptation Mechanisms

If network conditions vary and the current conditions are not suitable, one can basically take three strategies to cope with it.

First, one can simply wait until the data transfer is done anyway. This strategy works well if it does not really matter too much when the data actually arrives. A background download for a software update might be a good example where data integrity is very important but there are usually no strict timing requirements. Self-limiting the data rate for not time-sensitive communication (which naturally increases the duration a data transfer takes) may help to ease overall network congestion.

Second, one can try to send less data. As an example, a video surveillance camera could send a single picture every few seconds instead of a full video. This approach makes sense when outdated data has no value and it is better to have a reduced amount of data in time than the original data with a large delay. The reduced data rate might help to ease overall network congestion, too.

Third, one can try to transfer the data as efficiently as possible through e.g. compression or other technical approaches like proper packaging or appropriate protocol selection. These approaches can come with various effects, e.g. data loss in case of lossy compression or higher processing or memory requirements on the nodes.

Within TriCePS the following three approaches will be supported.

- *Time stretch* – If two nodes want to exchange data but there is no inherent urgency for the data to arrive at a certain time.
- *Payload adaptation* – If the *type of content* can be changed to adjust the data rate to the current network conditions.
- *Protocol adaptation* – Protocol selection and/or protocol parameter selection are used to react to the current network conditions.

Of course, a combination of several or all of these approaches is also possible. The goal of the TriCePS project is to develop the architecture and components necessary to support these adaptation mechanisms.

3 RELATED WORK

The TriCePS project touches several areas of research in computer networking, especially network measurement and monitoring, quality of service, transport protocols and application layer protocols.

3.1 Network Measurement and Monitoring

Generally, one can differentiate between active and passive measurements and monitoring (Mohan et al., 2011). With active monitoring, special probe packets are used to measure certain metrics. Active measurements create additional load. Passive measurements, in contrast, does not insert packets but use existing traffic and facilities to gain insights.

With TriCePS, we need continuous feedback on network state and potentially have to handle a huge number of connections. Due to the intrusive nature of active techniques, its bad scalability and infeasibility for continuous measurements on low-bandwidth links, passive techniques are used.

As a further constraint, we do not assume to generally have access to available passive network measurements on the path between two nodes (like router or switch SNMP statistics). We therefore have to use passive techniques that only rely on the end nodes of a connection, *passive, end-to-end network measurements* so to speak.

3.2 Quality of Service Metrics

The notion of quality of service (QoS) really has (at least) two relevant meanings for this project.

First, more general and highly related to service-level agreements (SLAs), a QoS attribute describes a measurable, specific aspect of the quality of a service. These aspects include metrics of performance, reliability and availability and SLAs then define QoS targets, e.g. 99.9% of I/O operations take less then 100 ms when measured over a one-minute sliding window.

Second, in packet-switched networking, QoS refers to a broad set of technologies that offer different types of service to different data packets, usually with the intent to provide guarantees with respect to network performance. QoS techniques include for example scheduling, resource reservation and traffic shaping.

While our generalized approach rules out both forms of QoS (the networks and devices are unknown a-priori so we cannot generally expect or enforce any type of SLA, neither can we expect or enforce the use of any of the mentioned QoS techniques), awareness of the term is important and many QoS metrics are directly relevant:

- *One-way delay* – Amount of time it takes for a packet from the sending node to the receiving node (in an end-to-end fashion).

- *Round trip time* – Delay from sender to receiver and back.
- *Delay variation* – Variation of packets' one-way delays.
- *Throughput* – Amount of data that is transferred in a certain period of time. Usually measured in bits per second.
- *Goodput* – Effective, application-level throughput without protocol overhead.

3.3 Application Level Optimization

With application level optimization, the application layer is informed about current QoS metrics and has the chance to adapt accordingly. This is not a novel approach; adaptive codecs have been around for quite a while.

The Adaptive Multi-Rate (AMR) audio codec, which is widely used in GSM and UMTS, is a prominent example. With AMR, different coding schemes can be used depending on link quality measurements performed on the receiver side (Ekudden et al., 1999). If conditions are good, AMR strives for speech quality (high speech bandwidth, low error protection). If conditions are bad, AMR strives for robustness (compromising on speech quality but boosting error correction and limiting bandwidth needs).

Another prominent example for existing application-level optimization is adaptive bitrate streaming (ABS) for video. ABS detects the user's current bandwidth and adjusts the quality of the video stream accordingly. One current ABS implementation is HTTP Adaptive Streaming (HAS). With HAS, segments of content (in the size of seconds) are encoded with different quality levels and made available over HTTP. The receiving side can then choose segments according to current bandwidth estimates. An interesting view into how YouTube is handling Quality of Experience for video can be found in (Sieber et al., 2016). NADA (Network-Assisted Dynamic Adaptation) (Zhu et al., 2013) also suggests adaptive real-time media applications in which adapt their video target rate and thus their sending rate based on explicit or implicit congestion signals.

In difference to the solutions above, TriCePS follows a more general approach by allowing the type of content being transported to change. This is supported by protocol switching and protocol parameter optimization mechanisms. However, approaches and solutions developed for by the other solutions such as NADA's delay estimations could generally be reused in TriCePS as components.

3.4 Protocol Switching

Protocol switching or, more specifically, time-independent communication protocol (re-)negotiation and switching enables two things: first and foremost, interoperability between a priori unknown end systems and second, further optimization of network usage through optimal protocol selection.

A recent, relevant work is Application-Layer Protocol Negotiation (ALPN), described in (Friedl et al., 2014). ALPN is a TLS extension and allows for protocol negotiation within the TLS handshake. The client sends a list of supported application protocols with its TLS ClientHello message. The application layer protocol to be used is then contained in the server's reply, the ServerHello message. For this, it is assumed that the server supports a number of protocols that are ordered by preference. A process similar to APLN will be implemented in TriCePS for protocol negotiation. The Session Initiation Protocol (SIP) with its Session Description Protocol (SDP) can be seen as another example for protocol negotiation (Rosenberg et al., 2002).

Finally, for a protocol repository an approach similar to software repositories will be used which manage and store applications and software packages that can be downloaded and installed on digital devices. Various software application or packet managers exist for specific operating systems (like Linux, Windows, macOS, Android) or programming languages (such as Maven for Java, PyPI for Python, CTAN for LaTeX).

3.5 Protocol Parameter Optimization

The parameter optimization for transmission protocols has a long history for TCP retransmission time out estimation (Paxson et al., 2011) based on heuristic formulas. (Balandina et al., 2013; Betzler et al., 2016) present the heuristic adaptation of the two CoAP parameters used for controlling the back-off mechanism retransmission timeout (RTO) and retransmission counter. Additionally the problem of missing ACKs for NON(-confirmable) messages leading to sparse RTT measurements is solved by artificial generated CON messages ("weak RTTs"). A new so-called Organic Network Control-based systematic learning approach is presented in (Kodama et al., 2008). The situation for a sending entity in the CPS is modelled as a competition between species in a harsh environment with the Lotka-Volterra competition model. The population of the species is mapped onto the TCP window size dependent on the received ACKs. Especially this work has high relevance for this project because its results were validated by a real implementation and experiments in a wide area network. The optimization of all ten TCP parameters for Web over TCP applications (Tomforde et al., 2015) has been developed to alter protocol configurations at runtime. It is equipped with online learning capabilities and safety considerations. The results are validated through simulation.

4 ARCHITECTURE

With the requirements and relevant related work being laid out in the former chapters, the proposed software architecture for TriCePS is presented.

4.1 Basic Design Decisions

- *Congestion-oriented* – Generally, the main network information required by TriCePS is whether there is congestion on the used network paths and the two main network-related metrics that are of importance for the purposes of TriCePS are latency and/or bandwidth. Changes in one or both of these two main metrics will serve as potential triggers for adaptations in the communication behavior of TriCePS-enabled CPS.
- *Socket-level mode of operation* – From a granularity point of view, all adaptation mechanisms are performed at socket level (as opposed to application-level or device-level). This is a comparatively non-invasive approach as each data flow can be treated separately and gives CPS users the means to prioritize individual data flows as necessary. Each application will also keep creating and controlling its communication sockets (as opposed to letting TriCePS create and control all sockets and passing the data back and forth to the applications).
- *Application-embedded* – From a deployment point view, the first version of TriCePS is embedded in the application process (as opposed to running in a separate process and accessed through inter-process communication mechanisms or being integrated in the kernel or network driver). This makes it easier to access socket information and modify socket settings if necessary.
- *In-band control* – TriCePS control signals are ideally be communicated in-band (i.e. over the same communication connection through which the payload is sent) to avoid creating additional connections. However, it remains to be investigated in the further course of the project if this can be correctly handled by applications and if there are any negative effects, e.g. caused by deep packet inspection security systems.
- *Non-collaborative* – Finally, the TriCePS adaptation mechanisms operates in a non-collaborative manner. At first, the Application Layer Optimization will choose the type/amount of data to be sent, then a suitable protocol is chosen, and finally a suitable set of protocol parameters is chosen (see figure 3). The application layer decisions will have the biggest influence on the overall communication and other adaptation mechanism will depend on these decisions.

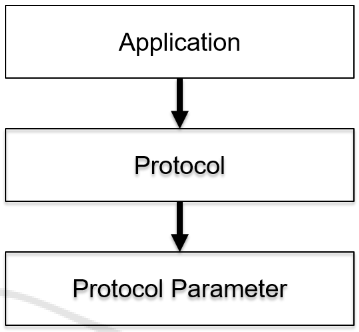

Figure 3: TriCePS Adaptation Components by Priority.

4.2 System Architecture Diagram

Figure 4 shows the TriCePS system architecture with TriCePS nodes, socket connections between those nodes and, for Node B, the internal TriCePS components.

Two arbitrary nodes of a communication system can be seen. Both nodes use TriCePS. Node A is only depicted as a black box while Node B is depicted with all the TriCePS components (Network Monitoring, Protocol Switching, Parameter Optimization) as well as all relevant building blocks of the overall system (the application itself, the TriCePS library, the network socket and the network). Note that the TriCePS functionality *Application Level Optimization* has no dedicated building block since it is implemented within the Application (which therefore relies on metrics gained by the Network Measurement component).

4.3 Components

The *Network Monitoring* component provides continuous feedback on network state (in the form of network metrics) to both the application and the Protocol Parameter Optimization component. It thereby uses passive, end-to-end network measurement techniques to obtain those metrics.

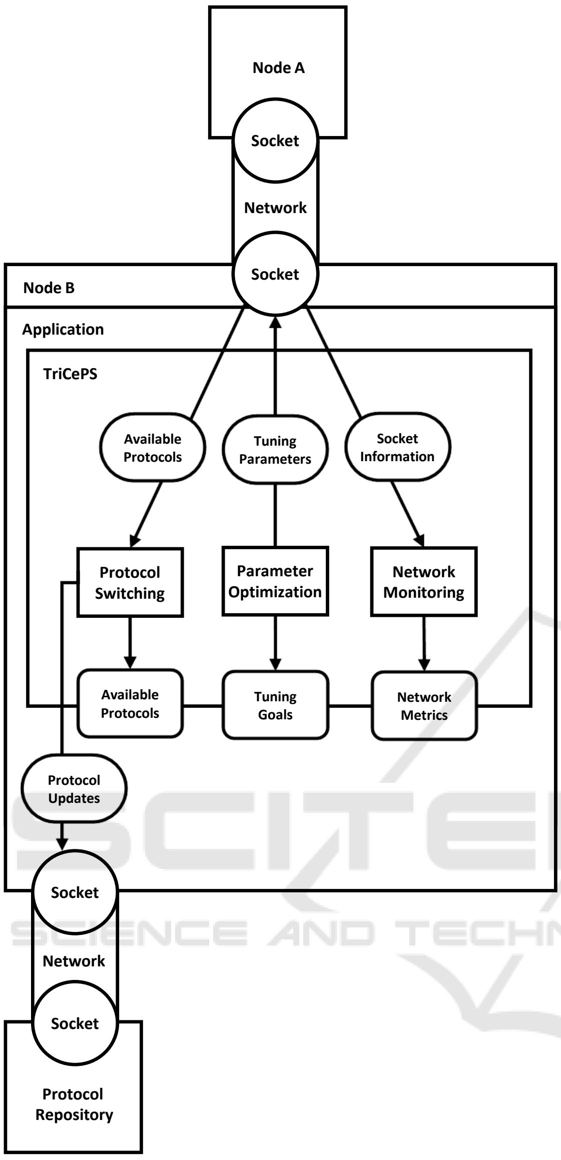

Figure 4: TriCePS nodes, components and interfaces.

The *Protocol Switching* component gathers information about the available protocols on the counterpart (node A) and the availability of new protocols on the protocol repository. It forwards this information to the application.

The *Parameter Optimization* component is set up by the application to work towards certain goals (e.g. low delay) and tries to tune certain socket parameters towards those goals. It does so without further triggering due to the application but solely on network metrics, socket parameters and mentioned goals. The work of the Network Parameter Optimization component is transparent to the application.

An application that uses TriCePS gets network metrics from the Network Management component, uses the Protocol Switching component to exchange information about available protocols (both at the counterpart and the repository) and assistance for protocol (re-)negotiation and sets goals for the Protocol Parameter Optimization component. It also controls the socket(s) used to send application data and thus the used congestion control algorithm.

5 DEMONSTRATION

For demonstration purposes the following scenario will be realized. A "smart surveillance camera" needs to transmit live digital footage over the communication network (see figure 5). The bandwidth requirements is significant while the amount of available bandwidth fluctuates. Tackling this problem through buffering is not sufficient as the footage needs to be transmitted and processed in real-time. To ensure liveliness of data, it can be better to compromise on video quality then to look at a stuttering and delayed high quality video. The amount of transmitted data will be adopted as shown in the following hierarchy (from good to bad network quality):

- HD video, normal compression, normal frame rate
- Individual images at low frame rate (<0.5 fps), reduced quality
- Individual image features (using image feature extraction)

The bandwidth requirements then vary between several Mbit/s (for HD video) and several hundred Bits/s (for textual descriptions of image features), that is a factor of 10000.

This demonstrator will serve as an example application and will make use of most or all TriCePS adaptations mechanisms and components at once. It will be adapted by industrial partner COPA-DATA for industrial use cases where for example near real-time availability of critical SCADA data is crucial.

6 CONCLUSION

Robust, interoperable, efficient and future-proof communication is essential for CPS. The architecture and components developed in TriCePS will provide a lightweight, practical solution for this purpose that does not rely on certain technologies or configurations of the network between two communicating CPS end nodes.

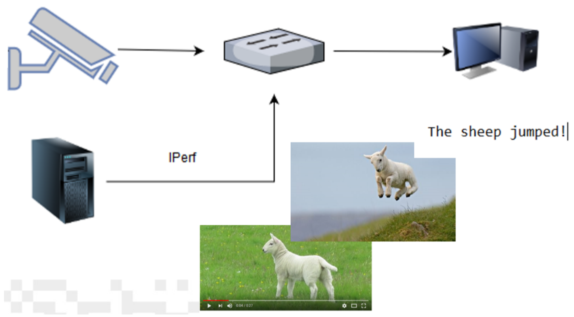

Figure 5: Depending on the underlying network conditions (controlled through the IPerf tool used to generate network load), the "smart camera" on the left side changes the application data that is sent. Either a full video stream, or only high resolution images, or textual descriptions of events and objects.

ACKNOWLEDGEMENTS

This project is partially funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" (https://iktderzukunft.at/en/).

REFERENCES

Balandina, E., Koucheryavy, Y., and Gurtov, A. (2013). Computing the retransmission timeout in coap. In *Internet of Things, smart spaces, and next generation networking*, pages 352–362. Springer.

Betzler, A., Isern, J., Gomez, C., Demirkol, I., and Paradells, J. (2016). Experimental evaluation of congestion control for coap communications without end-to-end reliability. *Ad Hoc Networks*, 52:183–194.

Ekudden, E., Hagen, R., Johansson, I., and Svedberg, J. (1999). The adaptive multi-rate speech coder. In *Speech Coding Proceedings, 1999 IEEE Workshop on*, pages 117–119. IEEE.

Friedl, S., Popov, A., Langley, A., and Stephan, E. (2014). Transport layer security (tls) application-layer protocol negotiation extension. RFC 7301, RFC Editor.

Kodama, M., Hasegawa, G., and Murata, M. (2008). Implementation experiments of tcp symbiosis: bio-inspired mechanisms for internet congestion control. *Proceedings of CQR 2008*.

Lee, E. A. (2015). The past, present and future of cyber-physical systems: A focus on models. *Sensors*, 15(3):4837–4869.

Minerva, R., Biru, A., and Rotondi, D. (2015). Towards a definition of the internet of things (iot). *IEEE Internet Initiative*, 1:1–86.

Mohan, V., Reddy, Y. J., and Kalpana, K. (2011). Active and passive network measurements: a survey. *International Journal of Computer Science and Information Technologies*, 2(4):1372–1385.

Paxson, V., Allman, M., Chu, J., and Sargent, M. (2011). Computing TCP's retransmission timer. RFC 6298, RFC Editor.

Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and Schooler, E. (2002). Session initiation protocol. RFC 3261, RFC Editor.

Sieber, C., Heegaard, P., Hoßfeld, T., and Kellerer, W. (2016). Sacrificing efficiency for quality of experience: Youtube's redundant traffic behavior. In *IFIP Networking 2016*.

Tomforde, S., Kantert, J., von Mammen, S., and Hähner, J. (2015). Cooperative self-optimisation of network protocol parameters at runtime. In *Informatics in Control, Automation and Robotics (ICINCO), 2015 12th International Conference on*, volume 1, pages 123–130. IEEE.

Weiser, M. (1999). The computer for the 21st century. *Mobile Computing and Communications Review*, 3(3):3–11.

Xia, F., Yang, L. T., Wang, L., and Vinel, A. Internet of things. *International Journal of Communication Systems*, 25(9):1101–1102.

Zhu, X., Pan, R., Ramalho, M., Mena, S., Jones, P., Fu, J., and D'Aronco, S. (2013). Nada: A unified congestion control scheme for real-time media. Draft IETF.