Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

Predictably Reliable Real-time Transport Services for Wireless Cyber-Physical Systems

Andreas Schmidt * **Pablo Gil Pereira** * **Thorsten Herfet** *

* *Saarland Informatics Campus, 66123 Saarbrücken, Germany*
(e-mail: {andreas.schmidt, gilpereira, herfet}@cs.uni-saarland.de)

Abstract: Cyber-physical systems increasingly leverage wireless networks for distributed control applications. In these systems, control and communication must find explicit *agreements* on the resilience and age-of-information (AoI) provided by the transport services to ensure stability. We present PRRT and its unique features to provide a predictably reliable real-time service that can fulfil these agreements. These features include cross-layer pacing, i.e. allowing an application to adapt to the system's bottleneck to achieve predictably low AoI. Finally, we highlight future directions for the transport service provided by PRRT with respect to its usage in constrained devices, where, e.g., energy demands play an important role.

Keywords: Communication Protocols, Communication Systems, Computer Networks, Transport Delay, Transport Properties

1. INTRODUCTION

Cyber-physical systems (CPS) (Lee (2008)) are increasingly using networked communication to fulfil distributed control tasks (cf. Baillieul and Antsaklis (2007)). Examples for this are highly automated manufacturing facilities, self-driving vehicles, and next-generation power grids. As these systems are both flexible and distributed, maintaining dedicated, proprietary communication infrastructure becomes an impediment. Therefore, these systems are increasingly using wireless communication standards, e.g. 5G cellular networks, 802.11ax local area networks, or low-power wide area networks. Wireless media is challenging for CPS, as they must cope with performance degradation due to mobility, interference, or intermittent connectivity.

Therefore, control and communication must adapt to this circumstance in a *cooperative* manner, i.e. taking the particular properties and assumptions of each other into account. This demands for explicit *control-communication agreements* that are formalized using, for instance, APIs. With the *terms of agreements* at hand, the transport layer in a communication system gets information it can use to tune or parameterize its various functions such as error and rate control. This paper presents the PRRT protocol that implements these functions and is—by exposing an API for control-communication agreements—able to offer a predictably reliable real-time transport service. Finally, we give future research directions with respect to transport layer services that are essential to control in next-generation wireless networks.

⋆ The work is supported by the German Research Foundation (DFG) as part of SPP 1914 "Cyber-Physical Networking" under grant number 315036956. We thank our project partners Stefan Reif and Wolfgang Schröder-Preikschat (Friedrich-Alexander-University Erlangen-Nürnberg) for the useful discussions and valuable feedback.

2. COMMUNICATION & CONTROL AGREEMENTS

In distributed and networked CPS, *agreements* must be made between the domains of control and communication—in order to ensure cooperative operations. While some control designs use jointly optimised and co-designed control and communication components, we consider designs that are loosely coupled to allow interoperability, flexibility, and abstract away implementation details.

2.1 The Need for Agreements (and Get-Out-Clauses)

Agreements between control and communication express under which transport conditions (i.e. quality-of-service) a control system's stability can be ensured. Making this explicit is essential, because control systems are often highly sensitive to delay and can start to oscillate if appropriate actions are not taken (cf. Branicky et al. (2000); Baillieul and Antsaklis (2007); Chwa et al. (2018)).

By knowing about explicit constraints of the controller, the transport layer can fine-tune its performance to fulfil this agreement, or clearly notify if the current channel conditions disallow this. While this notification looks, at first sight, like the transport layer is not committed to fulfil the agreement, this mechanism is essential as there will always be channel conditions under which a transport layer cannot fulfil the agreement. As the transport layer is responsible to inform the control layer, it is ensured that the layer that knows better about current and upcoming channel conditions (i.e. the transport layer) shares its information to serve the other (i.e. the control layer). In this *get-out* scenario, the controller has several options: (a) change to a controller that still maintains stability (e.g. by gain scheduling, cf. Leith and Leithead (2000)), (b) put the system into a safe state, or (c) fail as design assumptions do not hold anymore. Due to the better perspective

of the transport layer in comparison to the controller running on top, it is likely that these circumstances can be communicated earlier, giving more time to take countermeasures (making (a) and (b) more likely).

2.2 Conditions of Agreement

With this agreement architecture, the question is how an agreement between a controller and a transport layer stack can look like. For a controller, it is essential to state its demands about the delivery of messages with respect to:

Age-of-information (*AoI*) or information-freshness governs how much time is allowed to pass between sensing a physical quantity and processing it at the controller or between computing a control action and applying it through the actuator.

Resilience or tolerance w.r.t. packet dropout (*PD*) models how many messages a communication system is allowed to lose before the controller can no longer be confident that it can stabilize the physical system.

These two aspects can be modeled as follows: The *AoI* is a statistical distribution that the transport layer should fit within a given time window. A controller states the maximum tolerable AoI as well as other desired properties, such as median and latency spread. The spread of the distribution can in particular support control systems that are harmed by synchronization effects which a non-spread distribution would expose. The *PD* is modeled as the maximum number of messages that are allowed to be dropped in a fixed size sliding window—stating the robustness of the controller with respect to missing messages. An expired packet is typically considered as a packet dropout, so these two domains are correlated.

While these models have high fidelity, it is not straightforward to implement them in practice. The remainder of this paper considers that these two constraints are expressed in the form of AoI_{max} and PD_{max} (within a specific time window), providing a pragmatic solution to express the controller's requirements and expose them to the transport layer. In Sec. 4, we discuss how agreements can be more expressive than being scalar upper bounds.

3. TRANSPORT PROTOCOLS

To make networked systems fulfil these agreements and provide a service to controllers, it is straightforward to look at the transport layer. Agreements are useful because they (a) clearly state what the transport layer must ensure and (b) give headroom in stating what the transport layer does not need to ensure, i.e. it can transmit as slow as it wants, as long as it does not violate the agreement. For networked CPS, there are three essential network functions that must be ensured due to the nature of communication (in particular unreliability, cross-traffic, and increased delays): error, rate, and congestion control. While various implementations for these functions exist (e.g. for the TCP protocol), we propose two approaches: *Cross-Layer Pacing* and *Adaptive Hybrid Error Control*.

3.1 Cross-Layer Pacing

Leveraging the full potential of CPS requires interoperability and flexibility that are implemented, for instance,

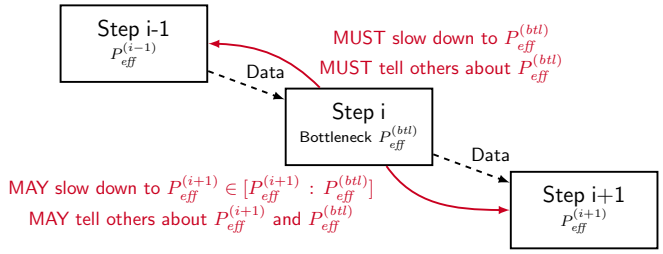

Fig. 1. Cross-layer pacing explicitly communicates paces to preceding and succeeding steps to attain just-in-time processing that ensures low age-of-information.

in statistically multiplexed networks—similar to the current Internet, whose technology is already used for edge computing scenarios (cf. Satyanarayanan (2017)). Thereto, these systems exhibit packet queues within host systems and at in-network nodes, which are needed for asynchronous process communication, such as the driver queue in Linux, or to cope with unexpected burst without packet losses. These queues bear the potential of self- or peer-incurred congestion—commonly known in the networking domain as *bufferbloat* (cf. Gettys and Nichols (2012)). This is clearly wasteful, especially because control information loses value while it is stored in a buffer and eventually becomes irrelevant due to an excessive AoI. This is a form of *waste*, considering a well-known definition of the term in the field of operations research (cf. Ohno (1988)). Information-processing systems with inherent buffering are wasteful for additional reasons, namely (a) buffer capacity must be allocated in the first place, (b) storing and retrieving from a buffer costs resources, and (c) buffers tend to be full, requiring drop-policies that cause more waste by discarding preprocessed units of work.

In this context, we have designed the *cross-layer pacing* approach that has been presented in Schmidt et al. (2019), which implements a form of rate control. This approach is based on two mechanisms: (a) continuously measuring the *pace* $P, [P] = \text{sec}$ of each communication and processing step and (b) continuously informing preceding and succeeding steps about this quantity. This information is shared as depicted in Fig. 1: Any step i instructs its preceding step $i-1$ to run no faster than the pace of i. Step $i-1$ computes the maximum of its own pace and i's pace and iterates by telling its preceding step, thereby adapting the chain in a way that the pace gets faster from first to last step. In such a scenario, buffers are only needed to compensate for imprecision in the synchronization process or unavoidable bursts, i.e. they only have to store a small number of packets (cf. Nichols and Jacobson (2012)).

Eventually, the scheme must slow down the first step—the application that generates data packets. In Fig. 2, we see how an unpaced sensor application produces many samples (every timestep of 1) but the network bottleneck only allows one message every 2 timestamps, thereby deteriorating the received signal at the controller and eventually producing losses due to buffer overflows in the network. When the sensor application is paced to the network (right half of Fig. 2), the signal is sampled and received appropriately. While cross-layer pacing provides the mean for the application to adapt to the slowest pace of the system, the selection of a sampling rate that optimizes

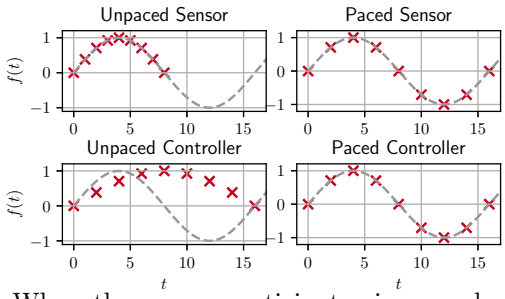

Fig. 2. When the sensor participates in cross-layer pacing, the sampling scheme is adapted to match the bottleneck of the system—achieving high fidelity and avoiding data loss due to buffer overflows.

the performance of the control algorithm is out of the scope of this paper. An interesting approach that optimizes the sampling rate given a real-time communication channel is introduced in Saifullah et al. (2014).

In order to achieve this, our approach provides an API with several means to ensure that the application takes part in the cross-layer pacing: (a) The application can become *pacing-aware* by querying the transport protocol for the bottleneck pace and adapting to it. (b) The application can passively take part, in being delayed by the underlying system. This requires no code changes and is transparent. (c) The application can use *synchronous send calls* that block until the next data packet can be created and sent. This approach requires periodic behaviour, as the transport layer measures the application and adapts its blocking behaviour to allow this just-in-time processing.

3.2 Adaptive Hybrid Error Control

Besides predictable timing, CPSs also demand predictable reliability, which can only be achieved if link characteristics (packet dropout rate, round-trip-time) and the application constraints (as defined by the agreements) are considered. This information is used to parameterize an *Adaptive Hybrid Automatic Repeat reQuest* (AHARQ) scheme (Gorius (2012)) to optimize the number of redundancy packets to be generated, which are then transmitted either proactively, using forward error coding (FEC), or reactively, using automatic repeat request (ARQ). The AHARQ scheme finds the optimal balance between FEC and ARQ as network conditions change. As the choice of coding parameters is further influenced by energy aspects and computing it is non-trivial, we are investigating learning-based approaches that allow to deploy AHARQ schemes on embedded devices with constrained power supply and computing resources.

3.3 Predictably Reliable Real-time Transport (PRRT)

Both cross-layer pacing and AHARQ support control-communication agreements and are implemented in the PRRT protocol. PRRT further supports agreements by taking a tolerable *AoI* and a tolerable *PD* as socket parameters. This has been analysed by Gallenmüller et al. (2019) to show how the reproducible timing behaviour of PRRT over wireless networks can support controllers.

First of all, PRRT is aware of the network and system conditions and hence able to notify the application in case the

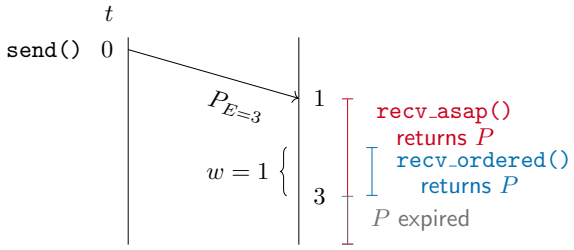

Fig. 3. Several receive calls support the different needs of controllers and govern, for instance, whether the expiry date (E) is used to delay the receiver-side delivery to the controller.

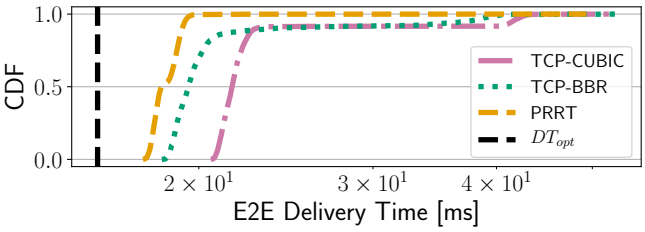

Fig. 4. Using cross-layer pacing, PRRT provides better communication characteristics than optimized TCP (cf. Schmidt et al. (2019)).

agreement cannot be fulfilled temporarily—allowing the control application to take appropriate actions. Additionally, PRRT's send and receive calls are designed in a way that they can support different applications with different communication approaches. Send-calls can, in combination with pacing, ensure that a controller can only send a packet when the end-to-end processing and communication chain is "free" (as governed by the pacing mechanisms), ensuring low age-of-information for the sent message.

PRRT further provides multiple *receive calls* to support different control strategies, as depicted in Fig. 3. `recv_asap()` delivers packets as-soon-as-possible, which makes channel-reordering visible to the application. Therefore, the tolerable AoI is only used to discard packets when the constraint cannot be met. `recv_ordered()` delivers packets that are close to their deadline, i.e. packets arrive at the application *in-time*. The application defines a window w to state how long before the deadline the packet is considered in-time, thereby controlling residual reordering.

Using PRRT's unique features, we are able to achieve delivery times that are predictably low, as can be seen in Fig. 4. The evaluation compares versions of TCP optimized for low-latency (i.e. with small buffers, disabled aggregation, and several other flags and options set) with PRRT. This has been carried out over the Internet ($RoundTripTime$ = 30 ms, $DataRate$ = 10 Mbit/s), where all transport variants are competing with other traffic and have to face queues that are induced by others. In private networks with less cross-traffic, PRRT is able to operate even closer to the optimum (not shown here).

The PRRT procotol is openly available [1] and comes with a C and a Python API that allow it to be used in various networked control applications. So far, PRRT is

[1] `http://prrt.larn.systems`

implemented on top of UDP in most scenarios, but PRRT's approaches do neither rely on UDP nor on IP. If a system implements routing/forwarding and process multiplexing, PRRT can be used to provide transport with predictable reliability and real-time characteristics. Finally, PRRT is analyzed using the X-Lap [2] tool for runtime, cross-layer, and intra-host analysis of transport stacks for latency and energy (cf. Reif et al. (2019b)). This tool reveals how much latency and jitter each processing step causes to messages.

4. FUTURE WORK

Control in wirelessly networked CPS is challenging as the used devices are fundamentally different from existing solutions. These devices have constrained computation, storage, communication, as well as power resources. These constraints demand system-aware solutions, i.e. that can fulfil agreements in spite of these constraints (Reif et al. (2019a)). Limited power supply is increasingly relevant as many control systems are mobile and energy costs are likely to increase in the next decades. Therefore, transport services for CPS must not only be latency- and resilience-aware, but also energy-aware. With this awareness, the next step is adaptiveness—allowing trade-offs between latency and energy, e.g. in the cross-layer pacing scheme by slowing down processing steps.

Furthermore, the agreements mentioned before only use scalar upper bounds on AoI and PD over a configurable window of time. Future work is going to consider models with higher expressiveness, i.e. using distributions instead of scalars and use confidence intervals around the distribution to model the area in which the actual distribution should fall to be acceptable to the controller. In addition, controllers may assign different values to different messages, so that a *value-of-information* metric can be considered that is depending on (a) the category of message and (b) the age of the message. As higher expressiveness comes with higher complexity, these approaches are going to be investigated with respect to practical, constrained devices on which future wireless control systems are built.

These challenges are investigated in the *Energy-, Latency-And Resilience-aware Networking* (e.LARN) project [3].

5. CONCLUSION

Upcoming networked CPS incorporate controllers that must increasingly deal with the unique features of wireless communication. Therefore, it is essential that the control and communication domains formalize *agreements* that describe under which communication conditions a system's stability can be ensured. The presented PRRT protocol leverages a first form of these agreements by considering the maximum age-of-information as well as maximum packet-dropout of a controller—and notifies the controller if these can no longer be fulfilled, allowing countermeasures to be taken. PRRT uses error and rate control to provide the adequate service to the controller; using unconventional approaches such as cross-layer pacing and adaptive hybrid error control, thereby providing better services to control applications than other transport protocols, such as TCP, even when those contenders are tuned for low-latency. Based on this, future PRRT versions are expected to support application constraints that go beyond single scalar values—allowing control engineers to better model their requirements towards the transport service.

[2] `http://xlap.larn.systems`
[3] `http://larn.systems`

REFERENCES

Baillieul, J. and Antsaklis, P.J. (2007). Control and communication challenges in networked real-time systems. *Proc. of the IEEE*, 95(1), 9–28.

Branicky, M.S., Phillips, S.M., and Zhang, W. (2000). Stability of networked control systems: Explicit analysis of delay. In *Proc. of the 2000 American Control Conference.*, volume 4, 2352–2357. IEEE.

Chwa, H.S., Shin, K.G., and Lee, J. (2018). Closing the gap between stability and schedulability: a new task model for cyber-physical systems. In *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 327–337. IEEE.

Gallenmüller, S., Glebke, R., Günther, S., Hauser, E., Leclaire, M., Reif, S., Rüth, J., Schmidt, A., Carle, G., Herfet, T., Schröder-Preikschat, W., and Wehrle, K. (2019). Enabling wireless network support for gain scheduled control. In *Proc. of the 2nd Intl. Workshop on Edge Systems, Analytics and Networking*, EdgeSys, 36–41. ACM, Dresden, Germany.

Gettys, J. and Nichols, K. (2012). Bufferbloat: Dark Buffers in the Internet. *Communications of the ACM*, 55(1), 57 – 65.

Gorius, M. (2012). *Adaptive Delay-constrained Internet Media Transport.* Ph.D. thesis, Saarland University.

Lee, E.A. (2008). Cyber Physical Systems: Design Challenges. In *Proc. of the 11th IEEE Intl. Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC).*

Leith, D. and Leithead, W. (2000). Survey of gain-scheduling analysis and design. *Intl. Journal of Control*, 73(11), 1001–1025.

Nichols, K. and Jacobson, V. (2012). Controlling Queue Delay. *Communications of the ACM*, 55(7), 42–50.

Ohno, T. (1988). *Toyota Production System: Beyond Large-scale Production.* CRC Press.

Reif, S., Gerhorst, L., Bender, K., and Hönig, T. (2019a). Towards low-jitter and energy-efficient data processing in cyber-physical information systems. In *Proc. of the 52nd Hawaii Intl. Conference on System Sciences*, HICSS. IEEE, Maui, Hawaii, USA.

Reif, S., Schmidt, A., Hönig, T., Herfet, T., and Schröder-Preikschat, W. (2019b). Δelta: Differential energy-efficiency, latency, and timing analysis for real-time networks. *ACM SIGBED Review*, 16(1), 33–38. Special Issue on 16th Intl. Workshop on Real-Time Networks.

Saifullah, A., Wu, C., Tiwari, P.B., Xu, Y., Fu, Y., Lu, C., and Chen, Y. (2014). Near Optimal Rate Selection for Wireless Control Systems. *ACM Trans. Embed. Comput. Syst.*, 13(4s).

Satyanarayanan, M. (2017). The emergence of edge computing. *IEEE Computer*, 50(1), 30–39.

Schmidt, A., Reif, S., Gil Pereira, P., Hönig, T., Herfet, T., and Schröder-Preikschat, W. (2019). Cross-layer pacing for predictably low latency. In *Proc. of the 6th Intl. IEEE Workshop on Ultra-Low Latency in Wireless Networks*, ULLWN. Paris, France.