Research Statement

Romain Jacob

My research focuses on **networked systems** and **communication protocols**. I would describe myself as a theory-inclined person with a strong taste for system design and implementation. More specifically, I am interested in designing systems with **formally-proven** performance or functional guarantees (*e.g.*, real-time systems) for which I benefit from my strong theoretical background. In addition, I like to think about simple yet efficient abstractions (*e.g.*, a communication primitive) to make complex systems easier to design, manage, and reason about. Finally, I like to question the status quo: it's when everyone thinks a problem must be addressed in a certain way that it is most interesting to try something different!

Following this mindset, I demonstrated during my doctoral studies that one can leverage network-wide flooding (*i.e.*, broadcast) in wireless networks to provide end-to-end real-time guarantees with high reliability and energy efficiency—whereas flooding is typically considered a wasteful mode of communication. This enabled the first-ever demonstration of closed-loop feedback control with stability guarantees over a multi-hop wireless network; a work that has received several awards from both academia[1] and industry.[2]

[1] 2019 ICCPS Best Paper Award and 2019 IPSN Best Demo Award
[2] Future Prize, Ewald Marquardt Foundation

In my future research, I am interested in applying similar concepts to **revisit how we design and manage computer networks**. Indeed, large scale computer networks are still managed a lot manually, which is getting increasingly harder. For example, intra-domain routing typically rely on assigning weights to network links and forward packets along shortest paths; link weights are tweaked by network operators to steer traffic along one or another path aiming to satisfy high-level network-wide specifications (*e.g.*, load-balancing, isolation, or waypointing) from the ground up which is inherently hard, error-prone, and the cause of many Internet outages nowadays. However, the recent availability of programmable networking hardware enables—among many other things—the implementation of efficient flooding primitives. I believe that, similarly to my wireless work, one can leverage such primitives to rethink routing protocol design and propose systems making large scale networks easier to manage while possibly providing unprecedented performance guarantees. This is one example of my vision for future networks, which I present in more details at the end of this document.

Research overview – Present and past

- During my doctoral studies, I worked on the design of **wireless communication** protocols for real-time cyber-physical systems; *i.e.*, distributed systems where sensing of and actuation on the world is interleaved with computations and communication.
- In addition, I investigated the problem of **replicability** in experimental networking research; more specifically, I proposed a concrete methodology for experiment design and data analysis that aims to foster the replicability of research results.
- Prior to my doctorate, I was trained and worked on more **theoretical problems**, which provided me with the formal background and mindset that shaped my later research.

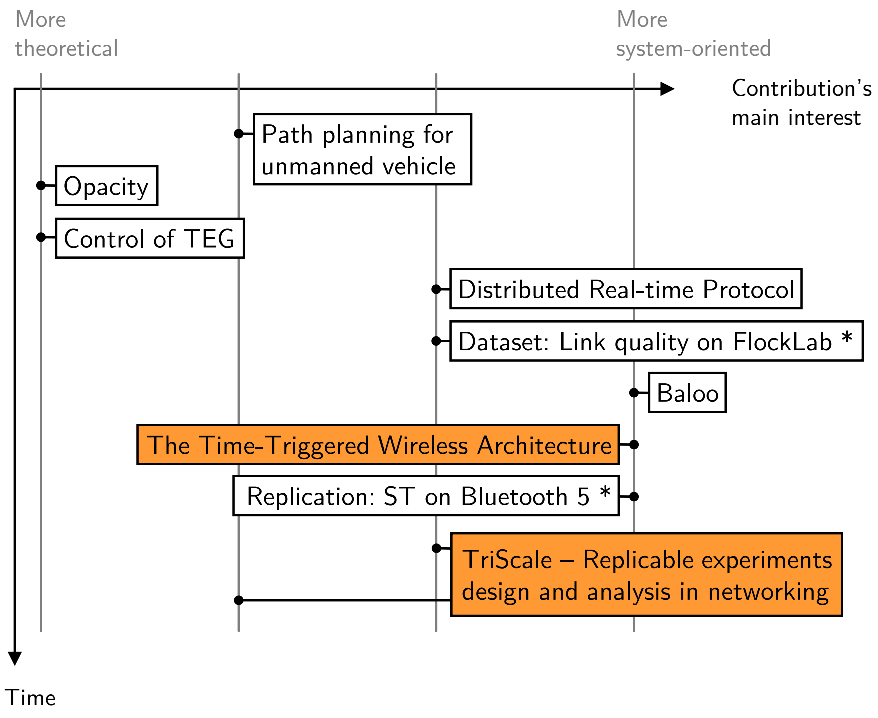
More theoretical
More system-oriented
Contribution's main interest
Path planning for unmanned vehicle
Opacity
Control of TEG
Distributed Real-time Protocol
Dataset: Link quality on FlockLab *
Baloo
The Time-Triggered Wireless Architecture
Replication: ST on Bluetooth 5 *
TriScale – Replicable experiments design and analysis in networking
Time

Figure 1: Overview of past and current research projects, oscillating between more system-oriented contributions and some of more theoretical interest. Highlighted projects are those I found most interesting. * indicates projects not described in this document.

Real-time guarantees in Wireless CPS

Many share the vision that our lives will soon be made simpler and more efficient by “smart” things, which concretely means embedding more and more sensing, computing, and actuation capabilities in distributed and networked devices. This concept is known as Cyber-Physical Systems (CPS), nowadays often marketed as the ‘Internet of Things’ or ‘Industry 4.0’. CPS are the foundation for countless applications in various domains such as smart manufacturing, building automation, fire protection, industrial process control, precision farming, personal assistance, or smart living and working environments.

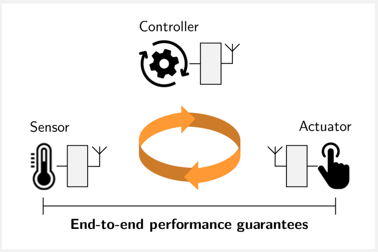

Figure 2: Distributed systems are often seeking to provide end-to-end performance guarantees, *i.e.*, between the execution of distributed tasks, such as the maximum time elapsed between the start of a sensing task and the end of the corresponding actuation.

In practice however, this vision is confronted with serious challenges, one of which is to provide reliable and energy efficient wireless communication between devices distributed within a house, a factory, or an entire city. Furthermore, many critical applications specify deadlines on their executions; guaranteeing to meet such deadlines is particularly challenging when considering (i) distributed CPS applications relying on multi-hop wireless communication and (ii) end-to-end guarantees between physically distributed tasks.

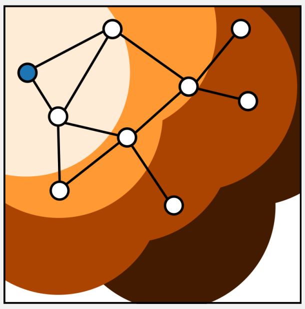
Figure 3: Basic illustration of flooding. The blue node (left) initiates the flood; receiving nodes repeat in the next time step. Each color corresponds to one time step, thereby capturing the wave-like dissemination of the original packet.

In 2011, researchers presented a new communication paradigm that would allow to address those challenges: *synchronous transmissions* (*ST*). *ST* was demonstrated to enable reliable and energy efficient communication in low-power wireless multi-hop networks. Put simply, *ST* allows neighboring devices to transmit their packets at the same time; in a classical setting, this would result in interference for the receiving devices which would then be unable to decode any packets. However, if the transmissions are sufficiently well synchronized (in the order of μs), two physical effects (constructive interference and the capture effect) make it likely for the receivers to decode one of the transmitted packets. In practice, this allows to build a very efficient multi-hop broadcast using a flooding strategy as well as provides μs-level time synchronization across the entire network.

In subsequent years, many wireless protocols have been proposed, tailored to different communication patterns, confirming the potential of the *ST* approach. However, these works remained mainly focused on wireless communication itself. In my work, I demonstrated that *ST* is a powerful tool to address the challenge of designing real-time CPS.

- I investigated the use of *ST* to design entire CPS providing end-to-end real-time guarantees and proposed two solutions: the *Distributed Real-time Protocol*, and the *Time-Triggered Wireless Architecture*, each optimizing for different performance objectives.
- I designed *Baloo*, a software framework facilitating the access to and adoption of *ST*.

The Distributed Real-time Protocol

Meeting end-to-end real-time guarantees inevitably requires the scheduling of distributed tasks to be coupled in some way. The Distributed Real-time Protocol[3] aims to couple them as little as possible. This system uses a contract-based traffic reservation scheme and an efficient reservation protocol (conceptually similar to RSVP). This protocol is integrated within a real-time analysis such as to support dynamically changing reservations without violating any real-time property of continuously transmitted flows.

[3] Jacob et al. 2016b

The main challenge of this approach consists in distributing the overall latency budget (*i.e.*, the end-to-end deadline) among the different components in the packet forwarding chain. We addressed this problem by performing a worst-case analysis of the contract establishment routine based on accurate models of all hardware and software components involved in the end-to-end packet forwarding. Albeit tight, our worst-case analysis yield suboptimal latency guarantees in the general case, as illustrated in Figure 4. That is the price to obtain timing guarantees while enforcing only loose couplings between distributed tasks.

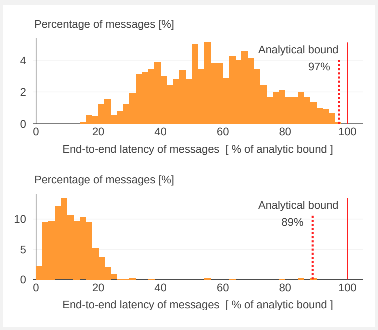

Figure 4: In simulation, one can trigger corner-cases to demonstrate the tightness of the worst-case analysis (top). However, in practice, such cases rarely happen which results in suboptimal latency in the general case (bottom).

The Time-Triggered Wireless Architecture

We thus proposed a second design exploring the other extreme in the design space: *i.e.*, minimizing the end-to-end deadlines one can guarantee thanks to a tight coupling of all communication and task executions across the entire system. This resulted in a design we call the Time-Triggered Wireless Architecture (*TTW*).[4]

[4] Jacob et al. 2020a

TTW statically co-schedules all task executions and packets communication together by solving a mixed-integer linear program. While this principle is well-known in a wired context, it cannot be trivially extended to low-power wireless: indeed, in order to save energy, communication is organized in rounds with no communication possible in-between — the radios are switched off. The mapping of packets to rounds creates additional scheduling constraints that happen to be non-linear. We addressed this problem with an approached inspired by network calculus: using arrival, service, and demand functions, we could reformulate those non-linear constraints into sets of linear ones.

The two drawbacks of static scheduling are its poor scalability and the lack of flexibility. We addressed both problems using an iterative approach. To retrieve some flexibility, *TTW* supports multiple modes of operation, between which the system can switch at runtime. Each mode has its own schedule, computed individually. The synthesis of a mode's schedule is also done iteratively by increasing the number of communication rounds in the schedule until a solution is found; hence, the first feasible schedule found is guaranteed to minimize the number of rounds used, and thus minimizing the energy consumed for communication.[5] *TTW* also guarantees that, although synthesized separately, the different mode schedules are compatible (*i.e.*, applications running in different modes have the same schedule in each mode). This is achieved by an analysis of the graph of possible mode changes resulting in the (provably minimal) set of constraints that must be added the schedule synthesis to guarantee mode compatibility.

[5] This is simply because each round induces some communication overhead.

We demonstrated that *TTW* can guarantee to meet end-to-end deadlines in the order of tens of ms using currently available low-power wireless devices. In a collaboration with CPS researchers, *TTW* has been used as the communication architecture to realize the first closed-loop control of inverted pendulums over a multi-hop wireless networks,[6] an achievement that won the ICCPS Best Paper Award and the IPSN Best Demo Award at CPS-IoTWeek 2019, as well as one of the 2019 Future Prizes of the Ewald Marquardt Foundation. More details about *TTW* can be found on the project webpage.[7]

[6] Mager et al. 2019, Baumann et al. 2019

[7] ttw.ethz.ch

Baloo– Facilitating the use of synchronous transmissions

While synchronous transmissions (*ST*) is a powerful technique which, as discussed above, allows meeting relevant end-to-end timing guarantees, it is difficult to use in practice. *ST* relies on tight control of the radio hardware which demands the skills of an embedded systems expert. To make the technique more widely usable — *e.g.*, by CPS engineers — we designed a software framework called *Baloo*.[8]

[8] Jacob et al. 2019

Baloo provides a high-level programming interface to design communication protocols based on *ST*. This allows the non-expert to specify the communication protocol logic (*i.e.*, when each device sends its packets) without worrying about the low-level radio control, all handled by *Baloo*. The framework is implemented as a time-triggered cooperative multi-threaded software; the execution is driven by communication rounds, composed of dedicated time slots where a *ST* primitive is executing. As time synchronization is key for *ST*, these slots are tightly scheduled using rtimers and run in interrupt context. The interaction with the protocol logic happens via callback functions executed between each communication slots to handle the packets (*i.e.*, passing on packets to send and processing the received ones). More time-consuming operations such as sampling a sensor or computing a control input are executed in dedicated protothreads that run between the communication rounds. While conceptually simple, a robust implementation of this design is non-trivial and was made possible by the 10+ years of experience of the research group with *ST*.

The main novelty of *Baloo* is to allow the use of multiple *ST* primitives within the same communication protocol. Indeed, multiple “flavors” of flooding have been proposed, which we call communication primitives: some more energy efficient, some more reliable, etc. For the first time, *Baloo* provides the ability to switch easily between multiple primitives at runtime and thereby significantly opening up the communication protocol design space; an opportunity that has already been taken by other researchers.

Baloo’s key contribution is to provide a programming interface that is flexible, easily portable to multiple hardware platforms, and induces limited performance overhead. We demonstrated that these goals were achieved by re-implementing several existing protocols and providing *Baloo* implementations for two widely used embedded platforms. Most importantly, a tool must be usable; we illustrated *Baloo*’s usability by having two Master’s students participating in the 2019 international EWSN Dependability Competition. Within a few weeks of part-time work, and without any prior knowledge about *ST*, each of them designed their own dependable protocol—a challenge thus far accessible only to embedded systems experts.

Performance guarantees in experimental evaluations

In most of today’s networking research, experiments are not performed, analyzed, and reported in a way that allows to replicate them; this problem is generally known as the “reproducibility crisis” and affects many scientific fields.

To address this problem, I investigated how one could foster and formally specify replicability in networking research, where it is often impossible to replicate the exact same experimental conditions, thus leading to unavoidable performance variability in general. We observed however that, albeit unavoidable, performance variability is not arbitrary. In particular, we proposed that the variability in networking measurements can be classified hierarchically using three different time scales which we call runs, series, and sequels.

1. The performance metrics of a system (*e.g.*, the power draw of an embedded device or the throughput of a TCP flow) vary over the execution of the system, which we call a **run**. Typically, the performance for the entire run is summarized using some statistic (*e.g.*, the median, the mean, the 95th percentile).
2. As variability is expected, one performs the same experiment multiple times (*i.e.*, a set of runs) which we call a **series**. Each run is the series yields a different performance metric value; the overall performance across the entire series is then typically summarized using the mean value.
3. Finally, consider someone attempting to replicate an experiment at a later point in time; the repetition of the entire series of run is what we call a **sequel**. In most cases, the overall performance metric value for the sequel differs from that of the original series, resulting in a third timescale of performance variability.

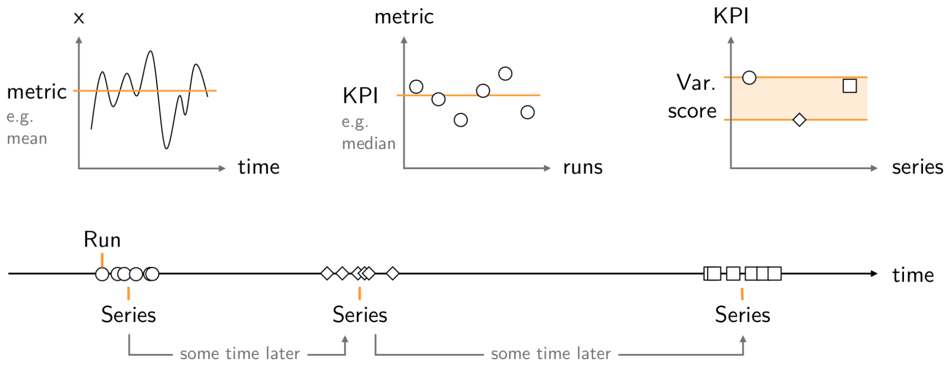

Figure 5: In networking experiments, performance often varies along three different timescales: during the run of the system, across multiple runs, and over longer periods of time. To address the replicability of the results, one must be aware of and account for these different sources of variability.

Based of that observation, we ask two hard questions:

Q1 Given such unavoidable variability, how can we **formalize replicability** and assess whether experimental results are indeed replicable?

Q2 Intuitively, the more repetition one performs, the more reliable the results are. But how can we **quantify** the relation between experimental effort (*i.e.*, the number of repetitions performed) and the confidence in the result? How can we rationally decide how many repetitions are enough?

We proposed answers to these questions with *TriScale*, a framework guiding experiments design and data analysis aiming to foster replicability of performance evaluation in networking.[9] *TriScale* applies appropriate statistical methods to account for the variability of each timescale and provides performance results with quantifiable levels of confidence (*e.g.*, a 95% confidence interval for the median throughput of a flow). The key idea of *TriScale* is to link these methods with the desired levels of confidence to derive the minimal number of repetitions that are required. This allows to answer **Q2**.

[9] Jacob et al. 2020b

Our research illustrates that the formalization of replicability (**Q1**) cannot be set in absolute terms or using binary criteria; some systems naturally exhibit more performance variability than others, thus it is natural to see more variability in experimental evaluations of such systems. Instead, we argue for a **quantification** of the replicability of experiments; that is, estimating the range of expected variation. It is then up to the different research communities to define acceptable ranges for a specific classes of systems (*e.g.*, congestion-control schemes, multi-hop wireless communication protocols, etc.).

TriScale is implemented, operational, and open source. The framework guides the user through a systematic experiment design and data analysis approach; based on the metrics of interest and desired levels of confidence *TriScale* computes the minimal numbers of runs, series, and sequels required. Then, given the collected data, the framework computes the appropriate statistics and returns performance results which are guaranteed to hold with the level of confidence specified.

iotbench.ethz.ch

My work on fostering replicability in experimental evaluations has been triggered by my involvement with IoTBench, a community-driven effort aiming to establish benchmarks for low-power wireless communication. In this context, we organized a series of international workshops[10] to raise awareness of replicability issues in our community as well as discuss practical solutions—such as *TriScale*.

[10] cpsbench2018.ethz.ch
cps-iotbench2019.ethz.ch
cps-iotbench2020.ethz.ch

Theoretical foundations

Before my doctorate, I worked on more theoretical problems such as path planning algorithms, discrete event systems verification, and synthesis of Petri Nets controllers.

Going either left or right – Path planning as a binary decision

During my Masters program, I spent one year as a visiting scholar at UC Berkeley in the group of Prof. Karl Hedrick where I worked on path planning algorithms for autonomous vehicles.

In free space, the shortest path towards a destination is a straight line, naturally. Now observe that in 2D space, if there is an obstacle on that trajectory, from a high-level view, you have only two choices: go around the obstacle from the left or the right side. Once the obstacle is walked around, you can go straight again. Using this idea and assuming knowledge of the map, one can turn the entire path planning problem into a sequence of binary decisions; this resulted in an algorithm called *directPath*.[11] This approach does not guarantee to find an optimal path but it yields better scalability properties[12] than the state-of-the-art A^* algorithm;[13] furthermore, *directPath* quickly detects unfeasible problems (*i.e.*, when there exists no paths from the source to the destination) whereas greedy algorithms have to exhaust the entire reachable state space, which can be very large. These properties make *directPath* an interesting candidate to quickly derive a feasible high-level plan which can be used as input for more refined motion planning algorithms.

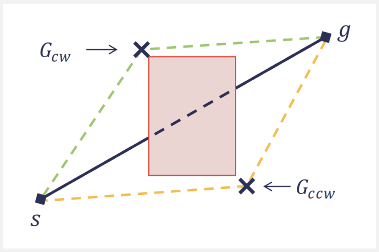

Figure 6: In 2D, there are only two ways to go around an obstacle: clockwise, and counter-clockwise. Each option can be used to define intermediate ‘sub-goals’ that one would have to traverse on its way towards the destination. That is the basic idea underlying *directPath*.

[11] Jacob and Hedrick 2014

[12] $\mathcal{O}$(number of obstacles) instead of $\mathcal{O}$(reachable states)

[13] A^* is essentially the Dijkstra algorithm extended with a heuristic giving the expected cost to reach the destination.

Opacity of discrete event systems

Opacity is an information flow property characterizing whether a given “secret” about a system behavior is hidden from an external observer. It is assumed the observer has full knowledge of the system’s structure but only partial observability. Based on its observations, the observer constructs an estimate of the system’s behavior. The secret is said to be opaque if the observer’s estimate never reveals the system’s secret. In other words, the system is opaque if, for any secret behavior, there exists at least one other non-secret behavior that looks the same to the observer.

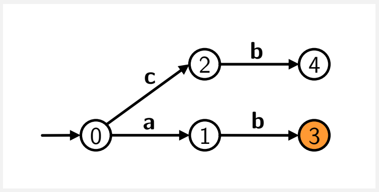
$$\begin{tikzcd} & & 2 \arrow[r, "\mathbf{b}"] & 4 \\ \arrow[r] & 0 \arrow[ur, "\mathbf{c}"] \arrow[r, "\mathbf{a}"] & 1 \arrow[r, "\mathbf{b}"] & 3 \end{tikzcd}$$

Figure 7: If 3 is the secret state (colored) and **b** is the only observable event, then this automaton is opaque. However, if **a** is also observable, it is not opaque as the observer is sure that the automaton is in state 3 after observing **ab**.

The notion of opacity comes in many flavors (language- and state-based opacity, probabilistic opacity, etc.) and there has been a lot of work in the past two decades investigating the validation and enforcement of opacity for various classes of discrete event systems. To help further progress, I compiled a survey of the state-of-the-art in that area in 2015; I highlighted some future research directions,[14] some of which have been pursued since.

[14] Jacob et al. 2016a

State feedback control for timed event graphs

Modeling timed behavior in discrete event systems is a difficult but important problem for many applications. One well-known approach relies on timed event graphs (TEG), a subclass of Petri nets, which dynamic behavior can be expressed using linear equations over the max-plus algebra. This allows, for example, to derive a controller enforcing timed properties such as “the system must stay in state X for less than one second.” This is reasonably easy when assuming that the system is fully observable (*i.e.*, the controller knows the timing of all the system’s state transitions), but this is often an unrealistic assumption.

During my Master’s thesis, I demonstrated that one can extend this approach to synthesize a realizable controller; that is, derive a control law from observable events only.[15]

[15] Jacob and Amari 2016

Future directions

The self-driving network vision

The ever-increasing complexity of networks and the criticality of the services that depend on them have made the vision of **self-driving networks** more and more popular; the idea is essentially to **specify what** the network must do, **not how** to do it.

This idea is not new and there is significant literature investigated it. The most notable one is the concept of software-defined networking (SDN) where the networking operations are logically divided between a control plane responsible for the routing process and a data plane that implements the forwarding state. In SDN, the control logic is centralized which allows to (re)compute an optimized forwarding state dynamically that is then pushed down onto the data plane. However, alike all centralized designs, SDN suffers from scalability issues that hindered its applicability and deployment.

With the emergence of **programmable networking hardware** in the last few years, one can envision a different approach to realize self-driving networks. Instead of a fully centralized control logic (like in SDN), we can now implement parts of the routing process directly in the data plane. As such, programmable hardware allows designing and implementing distributed routing algorithms that would flexibly (and autonomously) adapt the forwarding state to the network conditions at runtime. Using a distributed approach, one can hope to address the scalability problems that limited SDN.

I believe programmable networking hardware is ushering a new era in computer networks. While offering a still limited set of features, P4-programmable Tofino switches are readily available, affordable, and can perform—simple—computations directly in the forwarding plane while supporting multiple Gbps of throughput. The recent acquisition of Barefoot Networks, the manufacturer of Tofino switches, by Intel[16] is one more indicator of the serious potential of that technology at a large scale.

[16] Intel press release

Ultimately, programmable networking hardware might be to computer networks what microcontrollers have been to embedded systems: the **key enabler of a new technology**.[17] In the past twenty years, embedded systems researchers have developed means and methods to implement complex distributed protocols on highly resource-constrained hardware; I think there is an interesting parallel to be drawn with the limited capabilities of the Tofino. Given my background in both embedded systems and networking, I feel very attracted by this opportunity and I want to explore it in my future research. I describe some more concrete ideas in the following sections.

[17] In the embedded systems' marketing world, this is known as the "Internet of Things."

Let flooding shine—Self-building forwarding plane

One step towards the self-driving network vision would be to **let the network learn feasible routes** and populate its forwarding tables upon request. This would have two main benefits: first, it simplifies management, as one does not have to manually populate the tables; second, it allows keeping in memory only the entries that are effectively useful! Indeed, the ever-increasing number of IP prefixes is putting more and more load on the network devices, even though one may see traffic towards only a few prefixes.

One possible approach is to learn possible routes using **flooding**. Depending on the context, flooding is not necessary wasteful as its efficiency to disseminate information can pay off—as illustrated by my dissertation work. Moreover, flooding inherently discovers multiple feasible routes which offers an interesting premise for traffic engineering or fast-reroute. With programmable hardware, one can now implement flexible and efficient flooding primitives for layer 3 IP networks.

Using such a primitive, one can design a system where possible routes are not pushed from the control plane down to the data plane but where, instead, routes are discovered autonomously upon request in the data plane.[18] Core routers can independently learn the useful entries to store in their forwarding tables, and perform their forwarding decisions based on high-level control plane specification (*e.g.*, use least-congested path or load-balance traffic). I am currently working on this project as part of my postdoctoral research and our initial results are extremely promising.

[18] In spirit, this is similar to the discovery mechanism of Ethernet, but without the limitations of spanning trees.

Automatic network control

I think there is a lot of potential for research at the **interface of networking** and **automatic control**. The closed-loop feedback control over wireless project I was involved in (see "The Time-Triggered Wireless Architecture") has been a great example where we tailored the controller of distributed physical systems to match the communication performance provided by the network.

One can go one step further and consider the network itself as the system to be controlled. Can we design and control networks such that relevant traffic states (*e.g.*, throughput, link utilization, etc.) follow some target reference values? Concretely, the idea is to specify functional and performance requirements of the network behavior and co-design a networking protocol and control algorithm that would automatically steer the network towards the desired execution points. If we manage to map our networking problem to a more traditional automatic control problem, then one can leverage the wealth of results from control theory, similarly to what has been done many years ago for congestion control. Conducting such a project demands background a in both networking and control, which I happen to have.

More concretely, if one wants to apply, *e.g.*, model-based control methods, one challenge is that one needs a **network model**. In recent years, machine-learning techniques have been used to produce accurate models, however networks are notoriously volatile and there is no guarantee that a model trained on some data will remain valid or for how long. Together with one doctoral student, we are investigating an approach to detect when a network model is getting outdated directly in the data plane as well as strategies to "fix it." Results are very promising and will be submitted soon.

Benchmarking forwarding and routing protocols

> For better or worse, benchmarks shape the field—Patterson & Hennessy

The networking research community develops a lot of useful tools but, in comparison, very few benchmarks are available. This is problematic because without well-established benchmarks, it is really challenging to objectively compare different systems or algorithms, which is essential for the development of research literature. However, designing fair, representative, and practical benchmarks for networking research is not trivial, as I experienced first-hand through my involvement in IoTBench.[19]

[19] www.iotbench.ethz.ch
See "Performance guarantees in experimental evaluations"

One of the main challenges is to derive benchmark problems with good **generalization properties**; *i.e.*, ideally, based on the performance obtained on the benchmark, one can estimate the expected performance in other settings. Such generalization claims are difficult but possible to make with a good understanding of the underlying **statistical foundations**, which I became familiar with through my work on *TriScale*.

Concretely, one way to generalize is to define benchmark problems by random sampling from a "distribution of possible benchmarks." However, in networking, both the network (*i.e.*, the topology) and the scenario (*i.e.*, the traffic) are variables with statistical distributions, and not independent ones. A naive sampling across all possible topologies and traffic patterns is intractable, unpractical, and likely to produce unrealistic problems.

Instead, one practical approach would be to formalize the possible networks and scenarios by specifying a **domain-specific language** capturing realistic relationships between topologies and traffic patterns. Such an approach has been used successfully to generate synthetic datasets in the field of computer vision for autonomous driving. I would like to investigate these ideas further to develop computer networks benchmarks—given my other interests, I would naturally start with a benchmark for routing protocols.

Research Philosophy

As a researcher, some things are important to me regardless of the type of projects I work on. These essentially describe my research philosophy.

Quality over quantity Researchers are too often inclined or pressured to "publish one more paper." I think this is not good for people and not good for science. I much prefer working on fewer projects but dedicating the time necessary to make solid and meaningful contributions to the problem at hand. This tightly relates to my second point.

Research is not only writing papers I found the following quote extremely important for computer science and systems research.

> [A]n article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures—Jon Claerbout, as quoted by Buckheit & Donoho, 1995.

In other words, the main value of the research we produce is not the papers we write; it is the algorithms we develop, the systems we design, the experiments we perform. Consequently, it is essential that we do not limit ourselves to publishing papers, but that we also produce and release corresponding research artifacts.[20]

[20] One recent example is the release of a data visualization application that we published together with the accompanying paper and dataset: explore-st-data.ethz.ch

Open Science is just science done right I am a strong advocate of the principles known as "Open Science." I believe it is crucial that publicly funded research is shared for free and to everyone. We must also strive to make our research replicable by others in order to build confidence in science. I am glad to see these values spreading in the newer generations of researchers and I want to foster this in any way I can. At my own level, I made a public pledge to open science[21] outlining a number of rules I set to myself.

[21] romainjacob.net/pledge-to-open-science

Be proactive One obstacle to the normalization of Open Science in computer science is the lack of open access venues with a so-called diamond publishing model—*i.e.*, free to read and free to publish. Therefore, together with colleagues from several subfields, we are launching a new diamond open access journal: the Journal of Systems Research (JSys).[22] I believe this is a great example of researchers taking the lead to provide themselves with what they need.

[22] jsys.org

Collaborate I very much value collaboration, both within and outside of one's research group. Most of my own projects were collaborations, often with colleagues from other universities. Combining our different expertises has always been very beneficial in my research. As the systems we work on are getting ever more complex, I expect the need for collaboration in research to only increase.

For a complete list of publications, refer to my personal webpage.

Bibliography

Dominik Baumann, Fabian Mager, Romain Jacob, Lothar Thiele, Marco Zimmerling, and Sebastian Trimpe. Fast Feedback Control over Multi-hop Wireless Networks with Mode Changes and Stability Guarantees. **ACM Trans. Cyber-Phys. Syst.**, 4(2):18:1–18:32, November 2019. ISSN 2378-962X. DOI: 10.1145/3361846.

Romain Jacob and Saïd Amari. Observable Feedback Control of discrete processes under time constraint: Application to Cluster Tools. **International Journal of Computer Integrated Manufacturing**, August 2016. DOI: 10.1080/0951192X.2016.1224391.

Romain Jacob and J Karl Hedrick. directPath: An Expert Path Planning Framework to Handle Environment Knowledge. In **Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2014 IEEE 4th Annual International Conference On**, pages 7–12. IEEE, 2014. DOI: 10.1109/CYBER.2014.6917427.

Romain Jacob, Jean-Jacques Lesage, and Jean-Marc Faure. Overview of discrete event systems opacity: Models, validation, and quantification. **Annual Reviews in Control**, 41:135–146, 2016a. ISSN 1367-5788. DOI: 10.1016/j.arcontrol.2016.04.015.

Romain Jacob, Marco Zimmerling, Pengcheng Huang, Jan Beutel, and Lothar Thiele. End-to-End Real-Time Guarantees in Wireless Cyber-Physical Systems. In **2016 IEEE Real-Time Systems Symposium (RTSS)**, pages 167–178, November 2016b. DOI: 10.1109/RTSS.2016.025.

Romain Jacob, Jonas Bächli, Reto Da Forno, and Lothar Thiele. Synchronous Transmissions Made Easy: Design Your Network Stack with Baloo. In **Proceedings of the 2019 International Conference on Embedded Wireless Systems and Networks**, 2019. DOI: 10.3929/ethz-b-000324254.

Romain Jacob, Licong Zhang, Marco Zimmerling, Jan Beutel, Samarjit Chakraborty, and Lothar Thiele. The Time-Triggered Wireless Architecture. In **32nd Euromicro Conference on Real-Time Systems (ECRTS 2020)**, volume 165. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, May 2020a. ISBN 978-3-95977-152-8. DOI: 10.4230/LIPIcs.ECRTS.2020.19.

Romain Jacob, Marco Zimmerling, Carlo Alberto Boano, Laurent Vanbever, and Lothar Thiele. TriScale: A Framework Supporting Replicable Performance Evaluations in Networking. February 2020b.

Fabian Mager, Dominik Baumann, Romain Jacob, Lothar Thiele, Sebastian Trimpe, and Marco Zimmerling. Feedback Control Goes Wireless: Guaranteed Stability over Low-power Multi-hop Networks. In **Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems**, ICCPS '19, pages 97–108, Montreal, Quebec, Canada, 2019. ACM. ISBN 978-1-4503-6285-6. DOI: 10.1145/3302509.3311046.