My research develops theoretical foundations and practical tools for cyber-physical systems (CPS). These systems play an important role in many aspects of our daily lives; examples include medical devices, airplanes, cars, factory automation systems, intelligent buildings, and the emerging Internet of Things (IoT). Cyber-physical systems interact with the physical world and often perform life-critical functions, such as inflating an airbag in the event of a collision. When they fail, or even just respond too slowly, the consequences can be catastrophic. Hence, the goal of my work is to improve the safety, reliability, and security of cyber-physical systems.

Today’s cyber-physical systems are amazingly complex artifacts: for instance, a modern car contains more than 100 microprocessors that run thousands of software functions. As new features are being added and new technologies (such as multi-core processors) are being adopted, this complexity is only going to increase. Because of this trend, existing techniques for designing, analyzing, and verifying CPS – many of which were developed when systems still had a few dozen functions and perhaps one or two processors – are slowly reaching their limits. Thus, designing and analyzing CPS is becoming more expensive and time-consuming, and the number of safety flaws that are making it into deployed systems is increasing. This is evident, for instance, in the enormous number of cars and medical devices that have to be recalled every year. To add to the problem, many cyber-physical systems today offer very poor security, and we can regularly read about high-profile failures in the news: for instance, the botnet that compromised devices such as thermostats and baby monitors and used them to attack other systems, or the car that researchers compromised over its cellular link and drove into a ditch. With self-driving cars already on the horizon, computing is clearly becoming more closely connected with the physical world, but our current technology is not yet ready for the new challenges that this development brings.

Research focus. I have been developing theoretical methods, analysis tools, and practical design techniques that can be used to build large and complex CPS that are provably safe and secure. There are several ways in which size and complexity can create fundamental challenges for CPS design: (1) the system can simply have too many components, so existing analysis techniques are no longer practical; (2) the components can change their behavior at runtime, perhaps by running different tasks in response to changes in the environment; (3) new hardware features, such as shared caches or virtualization, can introduce new behaviors that the theoretical models do not capture; (4) the integration of CPS with large-scale distributed infrastructures, such as cloud platforms, introduces a host of new challenges, such as performance variability, hardware and software bugs, and cyber attacks; and (5) the number of ways in which the system can be compromised by an attacker increases dramatically. To address these challenges, I have organized my work into five main research thrusts:

- **Compositional theories** to build scalable analysis methods and platforms that can handle large and complex CPS;
- **Multi-mode techniques and design tools** that can guarantee safety even when the system changes its behavior;
- **Co-design methods** for making system development efficient, cost-effective, and correct by construction;
- **Trustworthy real-time cloud infrastructures** that can support mission-critical and life-critical applications; and
- **Detection and recovery techniques** that can guarantee resilience against a broad class of faults and attacks while providing timing guarantees.

My work tightly integrates theoretical foundations, concrete system-building, and practical applications. For instance, I have built a real-time virtualization platform that is based on interface theories to provide strong timing guarantees, and I have pioneered overhead-aware compositional analysis methods and scheduling techniques that capture important aspects of the underlying hardware and software platform to guarantee safety in practice while also exploiting modern hardware features to improve efficiency and performance. My research frequently bridges multiple disciplines, including real-time systems, formal methods, control theory, networking, databases, and distributed systems. For instance, in several recent projects I have successfully applied concepts and techniques from the real-time systems domain to areas such as security and networking.

Impact. I consider real-time and cyber-physical systems to be my core area, and many of my results have been published at the top conferences in this field: RTSS [7,30–32,44,52], RTAS [6,11,12,14,25,26,28,34,36,39,50,51], ECRTS [35,40], and ICCPS [43]. However, since my work is interdisciplinary, I have also been publishing at conferences in...
embedded systems (EMSOFT [22, 38, 47], CASES [21], IoTDI [29]), design automation (DATE [13, 19]), operating systems (OSDI [8], Middleware [53], HotOS [10]), and networking/cloud (NSDI [45], INFOCOM [27], CLOUD [46], HotNets [9]). My work has attracted substantial interest from industry; for instance, I have ongoing collaborations with General Motors, Toyota, Intel, and Facebook.

Several of my papers have been nominated for best paper awards at top conferences such as RTSS and EMSOFT, and one paper has received an outstanding paper award at RTAS; also, my student Max won first place at the ACM Student Research Competition at SIGCOMM 2017. The RTDS scheduler from our RT-Xen prototype is now included in the official distribution of the Xen virtualization platform, which is used by cloud platforms worldwide. Our prototype has been used in the automotive industry – e.g., EPAM Inc., GlobalLogic Inc., and the OpenXT project (by Assured Information Security Inc. and BAE) – and it has attracted interest from General Motors, Toyota, and Lockheed Martin. Several research groups around the world are using our prototype and our toolset, such as Prof. Stephen Crago (University of Southern California), Prof. Giorgio C. Buttazzo (Scuola Superiore Sant’Anna, Italy), Prof. Kim Larsen (Aalborg University, Denmark), Prof. Axel Legay (INRIA, France), and Prof. Thomas Nolte (Mälardalen University, Sweden).

I have held several leadership positions in my research community. I have been an Executive Committee member of the IEEE Technical Committee on Real-Time Systems since 2016; this is the steering committee for the top two conferences in real-time systems (RTSS and RTAS). I have served as the track chair for RTAS 2018, as the topic chair for DATE 2015 and 2016, as the publication chair for ESWeek 2017, and as the PC chair for workshops on compositional theory (CRTS) and multi-mode systems (APRES). I regularly serve on the program committees of leading conferences in real-time, embedded systems (RTSS, RTAS, EMSOFT, ECRTS) and design automation (DAC, DATE, ICCAD), and I was elected to be a member of the ACM Future of Computing Academy for my research vision and contributions.

In the rest of this statement, I will summarize the past and current work in my group, broadly classified under the five themes described above. The first three sections provide an overview of research thrusts that I considered core real-time and CPS, while the last two sections highlight recent research thrusts (over the past two years) that bridge CPS with other domains, such as distributed systems, networking, and databases.

1 Scalable design and analysis of cyber-physical systems

Since CPS often perform life-critical functions, they are often formally analyzed to prove that they will not fail or exceed a maximum response time. However, CPS are becoming increasingly complex, due to two recent trends: (1) functions that used to run on separate processors are now deployed on a common platform, and (2) different subsystems are increasingly interconnected. These trends have many potential benefits, such as higher efficiency and lower cost, but they also create a complex web of interconnected functions that is too large to be analyzed with existing techniques. Hence, the goal of my first research thrust is to develop scalable counterparts to these analysis techniques that will work for very large CPS.

1.1 Compositional theory

One of my key contributions is a new compositional approach to scheduling and analysis. In this approach, a complex CPS is broken down into individual components, which are analyzed individually at first. Then, an interface is generated for each component that hides the component’s internal details and captures only a few essential properties, such as timing and resource requirements. If the interfaces are chosen carefully, it is then possible to combine them into larger interfaces that cover bigger and bigger subsystems, eventually yielding the properties of the system as a whole. This approach can be applied both horizontally (for distributed systems) and vertically (for systems that are scheduled hierarchically), and I have developed compositional theories for both.

Compositional theory for distributed heterogeneous architectures. I was the first to introduce compositional timing analysis for systems whose components are modeled by very different formalisms [32]. I have pioneered techniques for composing state-based and stateless subsystems – described by Event Count Automata [7] and arrival/service curves in Real-Time Calculus (RTC), respectively – into distributed heterogeneous systems, and this work has sparked a new line of research in the real-time embedded research community that is exploring the connection between formal methods (such as timed automata or Lustre programs) and RTC. I also showed in [6] that, using a feedback control mechanism, certain state constraints – such as back-pressure effects caused by blocking writes on finite buffers – can be analyzed using a stateless model, such as RTC. This surprising result is attractive because the analysis is compositional and can be done very efficiently, but it still provides high accuracy. This finding has paved the way for a novel approach to analyzing a broad class of systems with such state constraints, such as automotive [38], avionics and control systems [28], and
multimedia systems [18], and the paper with my generalized results for automotive architectures was nominated for the best paper award at EMSOFT 2010 [38].

One of the challenges of compositional analysis is to ensure tightness of the analysis. The RTC framework provides a very general abstraction for the timing behaviors of event streams and thus is applicable to a broad class of real-time and networking applications. However, this generality does introduce analysis pessimism that is difficult to overcome: for instance, the computation of the output event streams, a foundational result of the RTC framework, is known to be highly conservative, but so far nobody had found a way to improve efficiency. With collaborators at Uppsala University and Hong Kong Polytechnic, I have developed a novel analysis technique that fixes this problem for the first time, and thus makes the analysis substantially tighter [44].

**Compositional theory for hierarchical systems.** I have developed a range of novel timing abstractions and resource-aware interface composition techniques [5, 11, 24, 25, 39] that enhance current work on multiple fronts, including optimality (which minimizes resources), associativity (which enables incremental design), and efficiency; I have also found a way to generalize these techniques to probabilistic settings [3, 4]. I have extended results to compositional scheduling of mixed-criticality systems [40]; this work was the first to identify the need to provide “isolation” among high-criticality components and a certain degree of timing performance for low-criticality components, both of which are critical for the safety assurance of CPS but were completely ignored by prior work.

My approach is unique in that it can not only handle heterogeneity, it turns heterogeneity into an opportunity! With the growing complexity of CPS, there is likely no “one size fits all” model that works well for all application domains. Instead, my approach can integrate models and analysis techniques from different disciplines and different domains, and thus leverage the strengths of each model in the scenarios for which it is best suited. For instance, I have demonstrated that, using a combination of formal verification and min-plus/max-plus algebra for the interface analysis [31], it is possible to efficiently analyze systems with far more complex timing behaviors than that was possible before. Similarly, I have shown that a combination of real-time scheduling and traffic shaping (from computer networks) can substantially improve schedulability and efficiency [34].

### 1.2 Incorporating practical issues into the theory

Prior work on compositional analysis was purely theoretical: it assumed an idealized platform in which all overheads are negligible. In practice, however, there are many sources of nontrivial overheads – such as preemptions, cache effects, context switches, and interrupts – that can substantially interfere with the execution of tasks. Ignoring these overheads when analyzing a CPS is dangerous because it can lead to wrong results: the analysis may suggest that the system will always meet its timing requirements, but then, at runtime, the system can nevertheless miss important deadlines [39].

To bridge the gap between theory and practice, I developed a novel overhead-aware compositional theory [39] that incorporates the potential platform overheads into the analysis and the interfaces, and thus makes the analysis a lot safer. One interesting outcome of this work was the insight that, in compositional systems, certain types of overhead can accumulate in ways that cannot be accounted for by existing methods, such as WCET inflation. My student and I then developed a taxonomy of overheads and novel ways to account for them [39], and we extended the resulting theory to multi-core virtualization platforms [52]. The paper with our results was nominated for the best paper award at RTSS 2013.

### 1.3 Platform and tool support

To bring the benefits of compositional theory to users and practitioners, I have led the development of the open-source CARTS tool set [37], which implements all the new techniques we have developed. Together with students and colleagues at Penn and Washington University, I have built the RT-Xen compositional scheduling platform [1, 25, 47], which provides a foundation for real-time virtualization on modern hardware. RT-Xen has attracted attention from industry – for instance, we are working with colleagues from General Motors to extend and integrate it into automotive systems – and it has been used in the automotive industry, by companies such as EPAM and GlobalLogic. The RTDS real-time virtual machine scheduler from RT-Xen has also been used by several research research groups around the world, both in the US and in Europe, and it is now part of the official Xen distribution; thus, the benefits of real-time virtualization are now available to users of some of the largest commercial cloud platforms, such as Amazon Web Services, Aliyun, Rackspace Public Cloud, and Verizon Cloud.
2 Safe run-time adaptation and response via multi-mode theories

Many cyber-physical systems need the ability to adapt to changes in their environment. For instance, a self-driving car must adapt its behavior according to the physical environment (such as changing road conditions or unexpected behavior of other vehicles) to avoid accidents; similarly, an unmanned aircraft avionics system must adapt its configuration during sudden turbulence or aircraft system failures. Adaptation can involve launching new tasks, as well as changing or terminating existing ones; a key challenge is to guarantee that the system meets its timing requirements not only in the old and new configurations, but also during the critical transition period.

2.1 Multi-mode theories

To formally model and analyze the timing behavior and performance of adaptive systems, I have developed a multi-mode real-time calculus (MMRTC) [31]. By combining the expressiveness of automata with the algebraic and compositional features of RTC, MMRTC enables much more general systems to be analyzed than before, including, e.g., systems with complex bursty event streams and with both resource- and application-level mode changes. I have also generalized these results to settings with complex processing and scheduling semantics [30].

A critical component of any multi-mode system is the specific protocol used for executing mode changes, also known as the mode-change protocol. When the system changes from one mode to another, the set of jobs that are active can come from both modes; as a result, even if each mode is schedulable in isolation, timing violations can occur during the transition period. By enforcing a certain execution behavior of the system during a mode change – such as aborting certain jobs, or delaying the release of new jobs – the protocol can avoid or minimize the potential overload, and thus avoid timing violations. At the time I started this work, the real-time community had already developed a variety of mode-change protocols; however, these protocols were specified informally, and most of them lacked a detailed experimental evaluation. As a result, it was difficult to tell whether a protocol was safe, or which protocol might be best for a particular system. To address this, I have pioneered a new, general approach. I have developed a unifying formal semantic framework (MCP) [36] that can be used to formally describe, analyze, and evaluate different mode change protocols; MCP can also be used to design and verify new protocols, and to derive optimal protocols for different classes of systems.

2.2 Multi-mode implementation platforms

The MCP framework enables, for the very first time, a systematic exploration of the design space for mode-change protocols. However, theoretical analysis and evaluation alone are not sufficient: platform overheads (which are usually abstracted away in the analysis) can have a big impact on the performance of a protocol, to the extent that tasks can miss deadlines even when the analysis predicts that they will be schedulable. To address this problem, my student and I have built SafeMC [26], the first system for efficiently designing, implementing, and experimentally evaluating mode-change protocols on hardware. Our key insight is that, while the existing protocols may appear very different at first glance, they actually have a lot in common. We identified a set of key primitives that operate on the smallest scheduling entity and encompass all mode-change behaviors; these primitives can be composed – at different levels of granularity – to form a protocol for the entire system. Thus, we can decompose existing protocols into 1) an protocol-specific algorithmic core and 2) a common, protocol-independent runtime system that executes the algorithm. This leads to important benefits: the algorithmic core can easily be customized or, in the case of a new protocol, be rewritten from scratch; the common runtime simplifies experimentation and enables fair comparisons on real hardware. The SafeMC paper received the Outstanding Paper Award at RTAS 2018.

2.3 Generalizations

Generalization to compositional systems. To enable compositional analysis of adaptive systems, I developed multi-modal real-time interfaces and composition techniques [33,35]. The connection between multi-mode and compositional reasoning that I introduced in this work stimulated further research on dynamic resource allocation techniques for component-based systems that rely on multi-mode analysis. Together with collaborators at General Motors, I am currently working on an application of multi-mode interface theories to automotive systems.

Generalization to multicore and mixed-criticality systems. In a modern CPS, tasks can have different levels of criticality; during a resource shortage, priority can thus be given to the more (safety-)critical tasks, while the less critical ones can be suspended or run at reduced capacity. Mixed-criticality systems have been studied in detail by the real-time community; however, the prior work had two important limitations: (1) it assumed that the criticality is always static,
which is unrealistic because, in practice, the criticality of a task depends on the current operating mode of the system; and (2) existing algorithms typically aborted all low-criticality tasks, which is dangerous because these tasks must meet a certain quality of service to ensure the overall safety of the system.

In my work, I have developed new mixed-criticality scheduling and analysis approaches that overcome both limitations. Together with a colleague at CMU, I have extended the multi-mode approach to mixed-criticality systems on multicore platforms [14] to allow for a much more precise descriptions of realistic systems. This work was the first to support changing the criticality levels at runtime, which is critical for real systems but also makes the analysis a lot harder. We also developed the first multi-mode mixed-criticality scheduling algorithm, as well as an implementation on multicore hardware. We further extended our results to the pipeline setting [13]; this work was the first to apply mixed-criticality scheduling and analysis to a distributed environment.

To enable a more fine-grained treatment of different criticality levels and to guarantee safety, my student and I, together with colleagues at Penn and KAIST, have developed a novel run-time scheduling approach [22, 23], along with its theoretical bounds, that can naturally adapt scheduling decisions to the dynamic behavior of the system. This approach not only utilizes resources much more efficiently, it also provides a better guarantee for multiple levels of criticality.

3 Co-design methods for cyber-physical systems

Today, it is common to separately design the control software of a CPS and the hardware platform on which it is to be deployed. This has two undesirable consequences: (1) the software and the platform must make conservative assumptions about each other, making the overall design more expensive than it needs to be, and (2) the software is implemented for (and verified on) an abstract platform model that ignores the more intricate details of the actual platform. Thus, properties that have been proven and verified can still be violated by the deployed system.

Control and platform co-design. I am working on efficient co-design methods that tightly integrate the development of control algorithms and platforms. The key insight is that, by making the controller aware of the properties of the platform, it is possible to simultaneously get better control properties and lower resource requirements. With collaborators from MIT and Toyota, I have developed adaptive control algorithms that can accommodate dropped signals, as well as novel platform mechanisms that efficiently implement various skip and abort strategies [42, 43]. We have applied our co-design methods to automotive CPS, and we have shown that they enable a much larger feasible design space while simultaneously reducing design cost by an order of magnitude, compared to existing methods.

Platform-aware model-based development. Model-based development is a way to build systems that are correct by construction. The software is built for a platform-independent model, which is then verified and, as a final step, translated to code for the target platform. However, this approach suffers from the weakness described above: the model typically abstracts away some details of the platform (such as I/O latencies and scheduling delays), and, as a result, the generated code may not actually have the properties that the verification appears to have proven. Together with students and colleagues at Penn, I am developing a “platform-aware” approach to model-based development that takes the interactions between software, platform, and environment into account when generating and verifying code [19–21]. We applied our initial results to medical CPS, such as infusion-pump systems, and we showed considerable improvements in safety. In our ongoing work, we are extending the approach to more complex settings, such as concurrency and distributed platforms.

4 Real-time cloud infrastructures

Cloud platforms have become immensely important in recent years. Public cloud platforms, such as Amazon Web Services, Google Cloud Platform, and Microsoft Azure, are now serving millions of businesses, and many large companies are running their own private cloud platforms to serve their internal workloads. Recently, there has been a growing trend towards using the cloud for critical infrastructure that traditionally is implemented on dedicated hardware, such as CPS and network function virtualization (NFV). However, current cloud technology is not a good match for such time-critical workloads: The cloud is shared among many different workloads, often by virtualizing the hardware and presenting each customer with his or her own virtual machines (VMs), which are then multiplexed on the underlying physical hardware. Virtualization is key for many benefits of the cloud, but it also increases variability: for instance, the performance of an application can vary considerably with the number and type of other applications that share the same physical machines, and latencies can increase, depending on how often a VM gets access to a physical CPU. While such latency variations can often be tolerated for many cloud workloads today, they are problematic for CPS and NFV, which require predictable timing. Variability also tends to have unpleasant consequences for security, especially on
publicly accessible platforms: for instance, a malicious VM can massively slow down other VMs that share the same physical machine with it. I was among the first to recognize this challenge, and I have since been developing scientific foundations for cloud-scale distributed systems with predictable timing.

4.1 Real-time virtualization

Virtualization-specific abstractions. As the first step towards a predictable cloud, my student and I have developed a real-time virtualization platform for modern multicore hardware that enforces isolation and predictability for multiple resources – not just for the CPU. Leveraging our prior work on compositional scheduling (Section 1), we developed new abstractions and analysis techniques for VMs that can ensure their tasks’ timing guarantees while tightly capturing the effects of concurrency enabled by the multiple cores and the complex interactions between tasks, virtual CPUs, and multiple levels of physical resources. Such interfaces provide a way for the hypervisor to allocate each VM the resources it needs to meet its timing constraints, and to enforce that allocation while eliminating potential interference.

We have also incorporated our results in both our CARTS toolset and RT-Xen prototype [47] to provide strong isolation and timing guarantees in multicore virtualization environments. Together with colleagues at Washington University, we integrated RT-Xen with OpenStack, one of the most popular cloud management systems, to support a mixed of time-critical and best efforts applications on the cloud [46].

Dynamic cache management for multicore virtualization. Real-time resource allocation also places some requirements on the platform itself: it must have a way to bound overheads, and it must be able to provide timing guarantees not only on computation but also on other types of resources, such as the shared cache and memory. To minimize interference from shared caches, my student and I have designed real-time cache allocation strategies [50] for OS kernels, based on a combination of page coloring and hardware mechanisms. We have evaluated our approach using concrete implementations for ARM and Intel processors; our results show that these strategies can completely remove cache interference between concurrent tasks, which makes the timing much more predictable.

Building on these results, we developed a new system for dynamic shared cache management on multicore virtualization platforms based on Intel’s Cache Allocation Technology (CAT) [51]. The core of our system is a novel approach to CAT virtualization that we call vCAT. vCAT is somewhat analogous to memory virtualization – for instance, it allows cache partitions to be mapped and preempted – but also has some interesting differences; for instance, the hypervisor cannot save the contents of a preempted partition. To enable practical use of our design, we provided a prototype implementation of vCAT on top of Xen, and we showed that our system can not only deliver strong cache isolation at both VM and task levels, it can also be configured for both static and dynamic allocations. This work brought forward the concept of ‘dynamic isolation’, which provides strong isolation while elastically scaling resources as are needed by the applications to maximize performance.

Holistic co-scheduling of multiple hardware resources. While earlier work has brought us closer to achieving timing isolation, most of it focused on allocation mechanisms for only two specific resources (CPUs and caches), and it did not address the allocation policy and analysis questions, e.g., about the right number of cache partitions to allocate to a task (or a core), or how to formally analyze the schedulability of the system. To bridge this gap, we developed a unifying framework that integrates shared cache allocation, memory bandwidth regulation, and CPU scheduling in a holistic and coordinated fashion. Using a prototype implementation on Xen and Intel hardware, we showed that this co-design of multi-resource allocation mechanisms and scheduling policies is highly effective in providing timing isolation among concurrent tasks and VMs, while also offering much better real-time performance and resource savings. Our work was the first to consider CPU, cache, and memory bandwidth allocation in a holistic manner in real-time multicore virtualization systems; the result is particularly interesting, as it opens many new challenges and opportunities for real-time co-scheduling of the complex layers of resources at cloud scales.

4.2 Scalable real-time cloud resource allocation

The above real-time virtualization platform provides a solid foundation for timing guarantees on virtualized nodes. However, moving to the cloud setting introduces a host of new challenges: Existing real-time resource allocation techniques are not practical for large, complex workloads, and they do not account for the complex topology of the network. Hence, my next step was to develop new allocation methods that can scale naturally using cloud/software interfaces. The key insight is that both the cloud hardware and the CPS/NFV software have a hierarchical structure. Hence, it is possible to use a compositional approach that is somewhat analogous to the approach from the first thrust: we can group both software components and cloud resources into larger and larger aggregates and describe their total resource supply or demand via interfaces. At each level of the resulting hierarchy, we can then use the interfaces
to perform resource allocation, and we can incrementally refine the interfaces as resources are requested or become available.

Based on this approach, my students and I have built NFV-RT, the first real-time cloud platform with strong timing guarantees for NFV chains [27]. We showed that this approach has a very small overhead, but it nevertheless provides timing performance guarantees and even supports dynamic reallocation of resources at runtime. We further extended our results to support complex DAG-based NFV applications and to enable adaptive responses to varying traffic rates of run-time requests [2]. Our evaluation, based on a prototype implementation the platform and real-world network traces and network functions, shows that our platform not only provides solid timing guarantees but also outperforms existing algorithms from the networking domain both in terms of worst-case and average latencies. These results, of which some appeared in INFOCOM 2016 and others are still under submission to RTAS 2019, provide a promising milestone towards time-predictable cloud platforms.

4.3 Performance diagnosis with temporal provenance

Moving the critical functionality to the cloud faces several new challenges beyond timing guarantees, among them the substantial increase in software and hardware bugs. Diagnosing cloud networks is a well-known challenge: with changing workloads, complex protocols, and a heterogeneous mix of devices, there are lots of interesting ways in which things can go wrong. Debugging networked systems is already difficult for functional problems, such as requests that are processed incorrectly; diagnosing timing-related problems, such as requests that incur a high delay, adds another layer of complexity: delays are often nondeterministic and can arise from subtle interactions between different components. Modern diagnostic tools offer little support, if any, for diagnosing problems related to timing.

However, the connection to timing offers us a new opportunity: the real-time community has already developed a comprehensive set of sophisticated tools that can be used to reason about timing behaviors, much of which are fundamentally similar to those shown in the networked setting. Recognizing this relationship, I have developed an interdisciplinary approach called temporal provenance [45], which will appear at NSDI 2019, that can help with diagnosing problems related to timing. Temporal provenance is a combination of research from the database/networking and the real-time domains: it is inspired by earlier work on provenance-based network debugging, but, thanks to new insights from real-time analysis, it can diagnose not only functional problems but also problems that are related to timing.

With students at Penn, I have developed a concrete algorithm that can generate temporal provenance for distributed systems, as well as a prototype debugger that implements this algorithm. Our experimental evaluation with real-world performance bugs based on real incident reports from Google Cloud Engine shows that our technique can successfully diagnose complex timing-related problems, while having comparable runtime overhead to that of existing tools.

5 Better security for real-time systems

The goal of my final research thrust is to improve the security of systems that rely critically on time. This problem has been getting far less attention than it deserves: the systems/networking and security communities have developed many security techniques, but usually with a focus on confidentiality or integrity (and not on timing or performance), whereas the real-time community has been focusing on timing guarantees, but less on malicious attackers. My work aims to bridge this gap, by taking my real-time expertise and by applying it to security problems in other domains.

5.1 Real-time responses to distributed denial-of-service attacks

Distributed denial-of-service (DDoS) attacks are a major threat to today’s cloud platforms. In a DDoS attack, the adversary compromises a large number of devices that are connected to the Internet, and then uses these devices to issue lots of requests to the cloud, which overload the attacked service. Since the malicious requests tend to be similar to the legitimate requests, these attacks are very difficult to defend against; the most common approaches are to try to recognize and drop the attack traffic somehow (which is difficult) or to simply use massive replication, which prevents the overload but can also be extremely expensive.

Together with colleagues and students at Penn and Georgetown, I have developed a radically different approach [9]: instead of trying to detect a specific attack, we leverage the modularity of modern OSes, libraries, and applications, and divide existing monolithic functional units into microservices that may be quickly spawned and replicated using real-time cloud resource allocation. We show that, by dynamically focusing replication efforts to match the specific target(s) of the attack, we not only gain resilience against a broad class of DDoS attacks but also maximize resource utilization while supporting critical, time-sensitive applications, which was not possible before. Our results have appeared at [9, 15, 16].
and our prototype has won first place at the ACM Student Research Competition at SIGCOMM 2017 [17].

5.2 Detecting performance problems and covert timing channels

When an adversary has compromised nodes in a classical distributed system, he can cause damage to the rest of the system through functional misbehavior – by causing the compromised nodes to do the “wrong thing”, or to fail to do the “right thing”. However, in a time-critical system, the adversary has a third option: temporal misbehavior, or doing the “right thing at the wrong time”. For instance, the adversary can cause the system to respond to a problem too late, or to perform a critical functionality too slowly. This is very difficult to detect.

Together with students and colleagues at Penn, I have developed a completely new approach to this problem, which is to detect temporal misbehavior through auditing. This is difficult because, unlike functional behavior, timing is difficult to reproduce; we addressed this by developing a new technology we call time-deterministic replay [8]. Interestingly, our solution also creates a new way to tackle covert timing channels, a problem the security community has been struggling with for decades. All earlier solutions to this problem were essentially mitigations that narrowed the channels; our approach is the first that can potentially close the channels completely. Our results have appeared at OSDI 2014 [8].

5.3 Bounded-time recovery

Since CPS often perform critical functions, e.g., in factory automation or in medical devices, a failure can result in physical destruction or loss of life. Hence, merely detecting an attack is not good enough. However, existing fault-tolerance techniques, such as BFT, are not a good fit for CPS: they involve massive redundancy, whereas CPS tend to be resource-constrained; they require strong assumptions, which are difficult to justify in CPS; and most of them avoid synchrony assumptions and thus cannot deliver the timing guarantees that are critical for CPS.

With students and colleagues at Penn, I have been developing a new approach to this problem that we call bounded-time recovery [10, 41]. Our key insight is that the physical part of most CPS has some inertia, which allows these systems to tolerate brief periods of undefined behavior, as long as the system returns to correct operation within a very short time (say, a few milliseconds). Thus, it is sufficient if the system can detect attacks and recover from them quickly. This approach is cheaper, requires fewer assumptions, and can provide the important timing guarantees. However, there is almost no work in this area: detecting problems within bounded time, reconfiguration under attack, and scheduling in partially compromised systems are all fresh problems that have received very little attention so far. These are the topics of my recently awarded NSF CAREER award, and we are planning to submit a paper with our initial results to ACM SOSP 2019 [41].

References


