Research Statement

Linh Thi Xuan Phan
http://www.cis.upenn.edu/∼linhphan/

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, smart grids, and the Internet of Things. 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, timeliness and security of cyber-physical systems.

Today's cyber-physical systems are becoming amazingly complex: a modern car already contains more than 100 microprocessors that run thousands of software functions. Since new features (such as AI) are constantly being added, this complexity is only going to increase. However, many classic techniques for designing, analyzing, and building CPS were developed at a time when systems were much simpler, with perhaps one or two processors and a few dozen functions. These techniques are slowly reaching their limits today: at current scales and with modern hardware, they can be slow, inaccurate, or even deliver results that are just plain wrong!

Research focus. My goal is to build a modern foundation for CPS that is ready for the new challenges that the increased size and complexity will bring. I have been focusing on four key areas: scalability, security, dynamic behavior, and support for modern hardware and cloud infrastructures. I have been developing compositional analysis techniques that can scale to large and complex CPS (Section 1); multi-mode techniques and design tools that can guarantee safety even when the system changes its behavior at runtime, perhaps in response to changes in the environment (Section 2); co-design methods that can take into account new hardware features, such as shared caches and virtualization (Section 3); technology for trustworthy real-time cloud infrastructure that can support mission-critical applications (Section 4); security techniques real-time systems (Section 5); and resource-efficient fault-tolerance techniques that can handle both functional and temporal faults (Section 6).

I use an end-to-end approach that starts with theoretical foundations but also includes practical system-building and concrete case studies, to verify that my solutions work as intended. I consider real-time and cyber-physical systems to be my core area, and many of my results have been published at RTSS [12, 33, 51–53, 66, 75] and RTAS [2, 10, 16, 17, 19, 26, 31, 32, 41, 42, 45, 47, 55, 57, 60, 71, 73, 74], the two top conferences in this field. However, my research also frequently builds bridges to other disciplines. As a result, some of my work has also appeared in top conferences in distributed systems [13, 14, 22, 23, 25, 27, 34, 43, 67, 68, 76], embedded systems [37, 38, 49, 50, 59, 69], robotics [4, 28, 29, 44, 46], design automation [18, 35, 70]), databases [63], and security [21, 48]. Several of my papers have won, or been nominated for, best-paper awards.

Impact. My work has attracted interest both from industry and from government agencies. For instance, the RTDS scheduler from our real-time virtualization prototype is included in the official distribution of the Xen virtualization platform, which is used by cloud platforms worldwide; our prototype has also been used in the automotive industry, e.g., at EPAM and GlobalLogic. Some of my recent research on fault tolerance and security was evaluated directly on spacecraft systems used for research and development at NASA, and has resulted in real impact in practice, such as the revision of the Time-Triggered Ethernet standard by TTTech and the adoption of our solutions at NASA and aerospace companies [48]. I am working with Roblox on ways to increase the scale and efficiency of their platform [34], and my work on a new synchronous architecture for data centers [15] has gained interests from GoogleX and SiTime. I also have ongoing collaborations with several companies, including Joby Aviation and Toyota.

Leadership. I have held several leadership positions in my research community. I have served on the Executive Committee of the IEEE Technical Committee on Real-Time Systems (2016–2019) and as the Secretary/Treasurer for ACM SIGBED (2019–2021). I have also served as a member of the Steering Committees for the top conferences in my field, including RTSS and RTAS (2016–2019), EMSOFT (2021–present), and RTNS (2022-present). I have served as the program chair for EMSOFT 2021, program co-chair for EMSOFT 2020, track chair for RTAS 2018, as the topic chair/co-chair for DATE (2015, 2016, and 2025), as the publication chair for ESWeek 2017, and as the PC chair for workshops on compositional theory (CRTS) and multi-mode systems (APRES). I co-founded and co-organized the workshop on computation-aware algorithmic design for CPS (2021–2024). I am an associate editor for ACM TECS and ACM TCPS (the leading journals in embedded systems and CPS, respectively), and I regularly serve on the program committees of leading conferences in real-time, embedded systems (RTSS, RTAS, EMSOFT, ECRTS, ICCPS) and design automation (DAC, DATE, ICCAD).

1 Scalable design and analysis of cyber-physical systems

Since CPS tend to 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. 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 [53]. I have pioneered techniques for composing state-based and stateless subsystems – described by Event Count Automata [12] 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 [10] 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 [59], avionics and control systems [45], and multimedia systems [30], and the paper with my generalized results for automotive architectures was nominated for the best paper award at EMSOFT 2010 [59].

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 [66].

Compositional theory for hierarchical systems. I have developed a range of novel timing abstractions and resource-aware interface composition techniques [9, 16, 40, 41, 60] 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 [7, 8]. I have extended results to compositional scheduling of mixed-criticality systems [61]; 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, by combining formal verification with min-plus/max-plus algebra, it is possible to efficiently analyze systems with far more complex behaviors than that was possible before [52]. Similarly, I have shown that a combination of real-time scheduling and traffic shaping (from computer networks) can substantially improve schedulability and efficiency [55].

Accounting for platform overheads. 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 [60].

To bridge the gap between theory and practice, I developed a novel overhead-aware compositional theory [60] that incorporates the potential platform overheads into the analysis and the interfaces, thus making 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 [60], and we extended the theory to multi-core virtualization platforms [75]. The paper with our results was nominated for the best paper award at RTSS 2013.

Prototypes 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 [58], 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, 41, 69], which provides a foundation for real-time virtualization on modern hardware. RT-Xen 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 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.

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) [52]. 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 [51].

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. Consequently, 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) [57] that can be used to formally describe, analyze, and evaluate different protocols. MCP provides a practical tool to design, verify and derive optimal protocols for different classes of systems.

To enable compositional analysis of multi-mode systems, I developed multi-modal real-time interfaces and composition techniques [54, 56]. 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 and forms a foundation for dynamic resource allocations on the cloud.

Multi-mode 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 [17], 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. This work won the Outstanding Paper Award at RTAS 2018.

Multi-mode scheduling for 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 [19] 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 [18]; 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, I have co-developed novel run-time scheduling approaches [38, 39, 61], 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.

Multicore resource allocation for multi-mode systems. While we have made advances in multi-mode theories and resource allocation techniques for shared multicore resources (such as last-level cache and memory bandwidth), no prior work has considered the multicore resource allocation for multi-mode systems. This problem introduces many new challenges that cannot be effectively solved by simply combining existing work from respective domains. In particular, since a multi-mode system can change tasks across modes, its resource demands change substantially from one mode to another. The question is can we dynamically adjust shared resources upon such changes to best match demands in each mode *without* jeopardizing schedulability during mode changes? In answering this question, my student and I developed Omni [33], the first end-to-end solution to this problem. Leveraging the hardware concurrency and the diverse resource demands of tasks, Omni intelligently co-distributes tasks and resources across cores to minimize overloads during mode transitions and maximize schedulability across all modes. We also implemented a low-overhead prototype of Omni in Litmus^{RT} and our RT-Xen prototype. Our work provides a first practical adaptive multi-core resource allocation technique for multi-mode systems that make the best out of the available hardware while ensuring timing predictability.

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. As cyber-physical systems become increasingly autonomous and resource hungry, there is a need for resource-efficient co-design techniques that can guarantee correctness, safety and timeliness, and that is robust to changes.

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. With students and colleagues at Penn, I have developed a "platform-aware" approach to model-based development that takes the interactions between software, platform, and environment into account when generating and verifying code [35–37]. We applied our results to medical CPS, such as infusion-pump systems, and we showed considerable improvements in safety.

Control-platform co-design and resource allocation. I have been working on efficient co-design methods that tightly integrate the development of control algorithms and platforms. Our first 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 [64,65]. 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.

Inspired by the above benefits, I spearheaded a new co-design approach for adaptive CPS. Our key insight was that the resource demands of a process can vary considerably during its execution; thus, there is an opportunity to improve utilization by dynamically matching the processes' resource demands with the hardware's supply. My students and I profiled tasks to measure their performance as we varied the amount of resources they were allocated; this enabled us to develop a rich timing model and to identify different phases with substantially different demands. We then designed an adaptive resource allocation algorithm that dynamically allocates resources at runtime, based on which task can benefit the most. This substantially increases throughput, decreases latency, and improves schedulability. Our solution [32] was the first to demonstrate the benefits of application-aware fined-grained resource control in multicore CPS.

With students and colleagues at UCSC, we further developed a novel method for predicting the time-varying probabilistic distribution of resource usage by the application based on the path structured multimarginal Schrödinger bridge problem (MSBP) [3,4]. While MSBP has been studied extensively in the theory community, to our best knowledge, our work presents its first application to the CPS setting. Our approach provides a promising direction to constructing accurate timing and resource models in complex CPS even with limited data.

Our model of control software resource uses further enables the development of DECNTR [31], the first controller and resource allocation co-design technique for adaptive CPS on multi-core platforms. Leveraging (i) a newly developed controllers that can safely switch between different implementations and sampling periods, and (ii) an efficient adaptive cache/memory bandwidth/CPU resource allocation mechanism, DECNTR is able to concurrently adapt control parameters and resource allocations to jointly optimizing safety, schedulability and robustness.

4 Trustworthy real-time cloud infrastructures

Cloud platforms have become very important in recent years; public cloud platforms, such as Amazon Web Services, Google Cloud Platform, and Microsoft Azure, are now serving millions of businesses. Recently, there has been growing interest in using the cloud for time-sensitive workloads, such as CPS control or network function virtualization (NFV); however, current cloud technology cannot support these workloads very well, since it relies heavily on virtualization and resource sharing. I was among the first to recognize this challenge, and I have addressed it by developing a foundation for cloud-scale distributed systems with predictable timing.

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 for just 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 level of physical resources [61,72]. 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 [69] 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 [68].

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 [73] 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) [74]. 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 [70, 71]. 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 holistically consider CPU, cache, and memory bandwidth allocation in real-time multicore systems; it won the Best Paper Award at RTAS 2019.

Scalable real-time cloud resource allocation. Giving timing guarantees for virtualized network functions is challenging because existing real-time resource allocation techniques are not practical for large, complex workloads, and because they do not account for the complex topology of the network. To address this, I have developed 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, so the compositional approach from Section 1 is applicable. My students and I have built NFV-RT, the first real-time cloud platform with strong timing guarantees for NFV chains [43]. 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 solution not only provides solid timing guarantees, it also outperforms existing algorithms from the networking domain both in terms of worst-case and average latencies. This work won the Best Student Paper Award at RTAS 2019.

Scheduling at microsecond scale. As cloud applications are becoming increasingly complex, consisting of functions that fan out to hundreds of backend servers, providing micro-second tail latency becomes critical to meet common SLOs. To achieve this, modern data centers typically run at low utilization, which not only wastes CPU resources but also provides no bound on the tail latency. With collaborators at Microsoft, I co-developed Perséphone [22], a new application aware, kernel-bypass scheduler that minimizes tail latency for applications executing at microsecond scales. Perséphone dynamically profiles the workload based on user-defined request filters to reserves cores and prioritize requests dynamically. Our insights are two-folds: First, as was observed in network congestion control, prioritizing short requests is critical to protect their service time from being delayed by long requests. And second, for workloads with wide service time distributions, leaving a small number of cores idle for readily handling potential future bursts of short requests can achieve lower tail latencies at higher utilization and can reduce the cores needed to serve the overall workloads. Our results show that this approach drastically improves requests tail latency while sustaining much higher load that the state of the art.

Performance diagnosis. Debugging networked systems is a difficult problem even without timing guarantees; trying to figure out why a certain action happened too late, or too early, or took too long, adds another layer of challenges, and current diagnostic tools do not offer much support for this. I have developed *temporal provenance* [67] as a way to help with this. Provenance has been used for diagnosing distributed systems before; my work adds a way to reason about timing, which raises a number of interesting additional challenges. My students and I developed both the concept of temporal provenance and a concrete algorithm that can generate it. We have built a debugger that uses provenance to diagnose both functional and temporal problems, and we have applied it to real-world performance bugs based on incident reports from Google Cloud Engine.

Quasi-synchronous platforms. Modern data-center hardware has fairly predictable timing – much more than what the common asynchronous model gives it credit for. I have been working on a next-generation data-center architecture [76] that leverages this property to get around some fundamental limitations of current algorithms. In a joint project with Roblox, I am using a similar approach to enable massive-scale virtual worlds [34].

5 Better security for real-time systems

Given the mission-critical nature of many CPS applications, good security is very important. However, this challenge 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 has been bridging this gap by taking my real-time expertise and by applying it to security problems in other domains.

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 [14, 20, 21]: 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. A prototype of this work won first place at the ACM Student Research Competition at SIGCOMM 2017 [24].

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 [13]. 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.

Real-time intrusion detection. Network intrusion detection is an extensive area of research in distributed systems. However, state-of-the-art solutions mostly rely on statically defined features of the traffic flow, making them less effective against new attacks. To address this, my group has explored unsupervised neural models to learn the expected network traffic that enables the detection of unknown novel attacks. An important challenge is to provide real-time detection: if the detection itself takes too long or an unpredictable amount of time, it can interfere with the application tasks, causing deadline violations or cause potential damage. We show that, using novel LSRM and convolutional autoencoder architectures, it is possible to achieve robust real-time network intrusion detection with high accuracy, even for unknown attacks [5, 6]. We are currently working on developing formal timing analysis to provide strong timing guarantees for learning-based detection methods.

Attack on timing isolation in the network. Modern CPS often use mixed-criticality networks such as Time-Triggered Ethernet (TTE) that allow critical traffic and non-critical best-effort traffic share the same switches and cabling. In such networks, the critical part of the system is *isolated* from the non-critical part, and thus non-critical devices cannot disrupt the operation of the critical devices. With collaborators from NASA and U. Michigan, we developed PCSPOOF [48], the first attack that can break TTE's isolation guarantees by attacking its synchronization. Our attack is based on two insights. First, it is possible for a non-critical best-effort device to infer private information about the critical part of the network that can be used to craft malicious synchronization messages. Second, by injecting electrical noise into a TTE switch over an Ethernet cable, a best-effort device can trick the switch into sending these malicious synchronization messages to other TTE devices. Our evaluation shows that attacks can succeed within seconds, and that each successful attack can cause TTE devices to lose synchronization for up to a second and drop tens of critical messages, both of which can result in the failure of critical systems like aircraft or automobiles. TTTech has revised the TTE standard in light of our findings, and NASA and several aerospace companies are implementing our proposed mitigations to avoid such an attack.

6 Real-time fault tolerance for CPS and robotic systems

Since CPS interact directly with the physical world, a failure can have disastrous consequences and lead to physical damage or even loss of lime. Hence, masking node failures – e.g., due to overheating, malfunctions, or interference – is very important. However, existing fault-tolerance techniques often are not a good fit for CPS: some are too expensive and do not work well within the tight resource constraints of a typical CPS; some assume crash faults, which capture only a subset of the things that can go wrong in a CPS; and some focus on ensuring functional correctness and ignore timing, which is not enough for CPS because the correct action, such as stopping an injection pump, can still be fatal if it is taken too late. To address this challenge, I have been developing alternative fault-tolerance techniques that *leverage* the unique characteristics of CPS, rather than seeing them merely as an obstacle that must be overcome.

Fast Byzantine fault tolerance by exploiting concurrency. The state-of-the-art defense for general attacks is to mask faulty nodes using Byzantine fault-tolerant (BFT) state machine replication (SMR). However, existing BFT SMR techniques require replicas to reach agreement on redundant input data before executing, thus adding at least $f+1$ rounds of communication latency to each execution on the input data to tolerate f faulty replicas. This high latency overhead is often unacceptable for CPS, which have stringent timing constraints. One way to reduce latency is speculation – forgoing agreement on inputs altogether and repeating executions when faults occur – but this approach is not suitable for CPS as it results in an even higher latency in the worst case.

With collaborators at NASA and U. Michigan, we developed a novel approach to reduce – or completely – avoid the latency overhead in BFT SMR by *exploiting the concurrency* that is inherent in today's CPS platforms. Our first result, IGOR [47], is a new speculative BFT SMR method that leverages multi-core processors common in modern CPS. Instead of agreeing on the data first, IGOR eagerly executes on data from redundant sensors in parallel on multiple cores; while these executions are underway, using a novel protocol, the replicas reach agreement on which execution will determine the system's final state. By parallelizing executions and agreement, IGOR substantially reduces latency and improves schedulability by up to 3.22×, and noticeably increases vehicle stability in spacecraft guidance, navigation, and control missions. Our second result, CrossTalk [49], exploits the redundant network planes and connectivity that already exist in modern CPS network to minimize latency (without requiring multi-core processors). Through an intelligent network routing protocol, CrossTalk completely removes the need for an agreement process but instead achieves formally-proven agreement as a mere consequence of the paths messages take through the network.

Bounded-time detection and recovery. Fast BFT methods such as above make it possible to build time-critical CPS that are robust against general faults and attacks. However, since BFT works by 'masking' faults, it still needs considerable amount of redundancy ($3f+1$ replica nodes to tolerate f faults), and it no longer work if the number of faults exceeds f. To solve this problem, I have developed an entirely new approach to CPS security, which we call *bounded-time recovery (BTR)* [15,26,27]. 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 within a bounded amount of time (as tolerable by the control design). This approach is much cheaper, requires fewer assumptions, and enables graceful degradation, while providing the important timing guarantees.

Realizing this approach is highly challenging, as BTR enters in uncharted territory: detecting problems within bounded time, reconfiguration under attack, and scheduling in partially compromised systems are all fresh problems that had received very little attention. Over the past few years, my research group has developed several novel interdisciplinary solutions to these research questions [11, 25–27], drawing insights from a diverse set of areas including security, distributed systems, multi-mode systems, and real-time theory. Our results provide a new, practical defense that is effective against Byzantine attacks and can offer provable guarantees. BTR has been applied to several practical case studies, including cars, railway control, smart grid, and multi-robot systems, and we are working with our NASA collaborators on potential adaptations to their systems.

Application to multi-robot systems. Multi-robot systems, consisting of dozens of interacting robots (or modules), are increasingly being deployed for important applications such as target tracking, warehouse logistics, and exploration. As with any other distributed system, the individual robots could malfunction or be compromised by an adversary. Although the robotics community has started to explore MRS-specific security solutions, current MRS solutions focus on specific, known attacks, such as Sybil attacks, certain types of lying in consensus protocols, or physical masquerade attacks. We developed ROBOREBOUND [25], the first general-purpose MRS defense for the fully-Byzantine threat model. As in BTR, ROBOREBOUND detects faulty robots and recovers from them in bounded time; however, we cannot directly apply BTR for two reasons: First, in an MRS, no robot can tell what another robot's sensors are showing, but this knowledge is necessary for detection. Second, a robot can cause damage simply by controlling its own actuators, so it is undesirable to simply ignore the faulty robot. RoboRebound solves this by using two very small and simple pieces of trusted hardware on each robot, which interpose on sensor and actuator communication. This provides a reliable way to obtain the correct inputs from another robot and to switch a compromised robot to a safe mode. We also developed a novel detection protocol for dynamic and unreliable environment, where robots may independently move in and out of communication range or packets may be lost due to unreliable networks.

Scalable real-time detection under actuator faults. In robotic systems, it is common for actuators to become faulty or degraded over time, which can significantly impact stability, control performance, and even cause crashes. State-of-the-art techniques for handling actuator faults are not only inefficient but also limited to only very small systems (e.g., with eight rotors or fewer). To address this challenge, my group has developed a novel approach for efficiently detecting and recovery from faulty rotors at run time in large-scale multi-rotor vehicles [44]. Our approach works by exploiting the invariance properties of roll/pitch error magnitude that we establish with respect to fault severity, planned trajectory and structure shape. In combination with a strategic search process and fault probe, we can quickly isolate the faulty rotor and accurately estimate its severity even while the system is in the air. Our evaluation on modular robots (ModQuad) further demonstrates that our technique enables prompt mitigation actions and thus helps improve the system robustness against future faults. To our best knowledge, our work is the first that scales to systems with several dozens of rotors. To translate our technology to practice, we have collaborated with Joby Aviation to extend our technique to vertical takeoff and landing vehicles (VTOLs), modeled after their own systems [29]. Using both fixed wings and multiple rotors for actuation, VTOLs provide many advantages over traditional fixed-wing aircraft and multi-rotor vehicles, and are an emerging class of robotic systems with successful real deployment in urban transportation [62]. Our group is working with collaborators at Joby Aviation to explore potential adoption of our technique to their systems.

Recovery via run-time self reconfiguration. To recover from actuator faults, we have developed mitigation techniques that adapt the controller onboard to counter the effect of the fault [29, 44], re-stabilize the system and maintain its mission. However, there exists another exciting opportunity: modern large-scale multi-robot systems, such as modular vehicles, have the ability to disassemble and reassemble to form different structures and shapes. Exploiting this agility, we pioneered a new approach to recovery that is based on self-reconfiguration [28, 46]. The idea is to reconfigure the structure of the system on the fly to minimize the impact of the faulty modules on the motion of the structure over the entire trajectory, for example by placing it closer to the center of mass. Realizing this approach is highly non-trivial for two reasons: first, we need to ensure the faulty modules are never isolated during reconfiguration to avoid crashes; and second, the entire reconfiguration process must be done in a timely fashion to conserve energy and reduce impacts on the trajectory/mission. Our solution achieves these goals using a novel a mixed integer linear program for optimal module-to-structure position allocation and an efficient dynamic programming algorithm that minimizes the reconfiguration cost. Our evaluation on ModQuad shows that our technique substantially increases the robustness of the system while utilizing resources efficiently. [46] was nominated for the Best Paper Award in Multi-Robot Systems at ICRA 2021.

References

[1] Real-Time Xen Virtualization (RT-Xen). `https://sites.google.com/site/realtimexen/`.

[2] S. Abedi, M. Demoulin, N. Gandhi, Y. Li, Y. Wu, and L. T. X. Phan. RTNF: Predictable Latency for Network Function Virtualization. 2019.

[3] G. A. Bondar, R. Gifford, L. T. X. Phan, and A. Halder. Stochastic learning of computational resources. *In submission.* `https://arxiv.org/abs/2405.12463`.

[4] G. A. Bondar, R. Gifford, L. T. X. Phan, and A. Halder. Path Structured Multimarginal Schrodinger Bridge for Probabilistic Learning of Hardware Resource Usage by Control Software. In *Proc. American Control Conference (ACC)*, 2024.

[5] N. Borgioli, F. Aromolo, L. T. X. Phan, and G. Buttazzo. A Convolutional Autoencoder Architecture for Robust Network Intrusion Detection in Embedded Systems. In *Journal of Systems Architecture (Accepted)*, 2024.

[6] N. Borgioli, L. T. X. Phan, A. Biondi, and G. Buttazzo. Packet-based Real-Time Intrusion Detection on Edge Devices. In *Proc. Real-time and IntelliGent Edge Computing Workshop (RAGE)*, 2023.

[7] J. Boudjadar, A. David, J. H. Kim, K. G. Larsen, M. Mikucionis, U. Nyman, A. Skou, I. Lee, and L. T. X. Phan. Flexible Framework for Statistical Schedulability Analysis of Probabilistic Sporadic Tasks. In *Proc. IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*, 2015.

[8] J. Boudjadar, A. David, J. H. Kim, K. G. Larsen, M. Mikucionis, U. Nyman, A. Skou, I. Lee, and L. T. X. Phan. Quantitative Schedulability Analysis of Continuous Probability Tasks in a Hierarchical Context. In *Proc. 18th International ACM SIGSOFT Symposium on Component-Based Software Engineering (CBSE)*, 2015.

[9] J. Boudjadar, I. Lee, J. H. Kim, L. T. X. Phan, K. G. Larsen, and U. Nyman. Generic Formal Framework for Compositional Analysis of Hierarchical Scheduling Systems. In *Proc. IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*, 2018.

[10] A. Bouillard, L. T. X. Phan, and S. Chakraborty. Light-weight Modeling of Complex State Dependencies in Stream Processing Systems. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2009.

[11] Y. Cai and L. T. X. Phan. GeoShield: Fault Detection and Recovery for Geo-Distributed Cyber-Physical Systems. *In submission.*

[12] S. Chakraborty, L. T. X. Phan, and P. S. Thiagarajan. Event Count Automata: A State-Based Model for Stream Processing Systems. In *Proc. IEEE Real-Time Systems Symposium (RTSS)*, 2005.

[13] A. Chen, W. B. Moore, H. Xiao, A. Haeberlen, L. T. X. Phan, M. Sherr, and W. Zhou. Detecting Covert Timing Channels with Time-Deterministic Replay. In *Proc. USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2014.

[14] A. Chen, A. Sriraman, T. Vaidya, Y. Zhang, A. Haeberlen, B. T. Loo, L. T. X. Phan, M. Sherr, C. Shields, and W. Zhou. Dispersing Asymmetric DDoS Attacks with SplitStack. In *Proc. ACM Workshop on Hot Topics in Networks (HotNets)*, 2016.

[15] A. Chen, H. Xiao, A. Haeberlen, and L. T. X. Phan. Fault Tolerance and the Five-second Rule. In *Proc. Workshop on Hot Topics in Operating Systems (HotOS)*, 2015.

[16] S. Chen, L. T. X. Phan, J. Lee, I. Lee, and O. Sokolsky. Removing Abstraction Overhead in the Composition of Hierarchical Real-Time System. In *Proc. IEEE Real-Time Technology and Applications Symposium (RTAS)*, 2011.

[17] T. Chen and L. T. X. Phan. SafeMC: A System for the Design and Evaluation of Mode-Change Protocols. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2018.

[18] D. de Niz, B. Andersson, H. Kim, M. Klein, L. T. X. Phan, and R. Rajkumar. Mixed-Criticality Processing Pipelines. In *Proc. Design Automation and Test in Europe (DATE)*, 2017.

[19] D. de Niz and L. T. X. Phan. Partitioned Scheduling of Multi-Modal Mixed-Criticality Real-Time Systems on Multiprocessor Platforms. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2014.

[20] H. Demoulin, I. Pedisich, L. T. X. Phan, and B. T. Loo. Automated Detection and Mitigation of Application-level Asymmetric DoS Attacks. In *Proc. ACM SIGCOMM 2018 Workshop on Self-Driving Networks (SelfDN)*, 2018.

[21] H. Demoulin, T. Vaidya, I. Pedisich, B. DiMaiolo, J. Qian, C. Shah, Y. Zhang, A. Chen, A. Haeberlen, B. T. Loo, L. T. X. Phan, M. Sherr, C. Shields, and W. Zhou. DeDoS: Defusing DoS with Dispersion Oriented Software. In *Proc. 2018 Annual Computer Security Applications Conference (ACSAC)*, 2018.

[22] H. M. Demoulin, J. Fried, I. Pedisich, M. Kogias, B. T. Loo, L. T. X. Phan, and I. Zhang. When Idling is Ideal: Optimizing Tail-Latency for Highly-Dispersed Datacenter Workloads with Persephone. In *Proc. ACM Symposium on Operating Systems Principles (SOSP)*, 2021.

[23] H. M. Demoulin, I. Pedisich, N. Vasilakis, V. Liu, B. T. Loo, and L. T. X. Phan. Detecting Application-layer Denial-of-Service Attacks with FineLame. In *Proc. USENIX Annual Technical Conference (ATC)*, 2019.

[24] H. M. Demoulin, T. Vaidya, I. Pedisich, N. Sultana, B. Wang, J. Qian, Y. Zhang, A. Chen, A. Haeberlen, B. T. Loo, L. T. X. Phan, M. Sherr, C. Shields, and W. Zhou. A Demonstration of the DeDoS Platform for Defusing Asymmetric DDoS Attacks in Data Centers. In *Proceedings of the SIGCOMM Posters and Demos*, 2017.

[25] N. Gandhi, Y. Cai, A. Haeberlen, and L. T. X. Phan. RoboRebound: Multi-Robot System Defense with Bounded-Time Interaction. In *Proc. European Conference on Computer Systems (EuroSys)*, 2025.

[26] N. Gandhi, E. Roth, R. Gifford, L. T. X. Phan, and A. Haeberlen. Bounded-time recovery for distributed real-time systems. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2020.

[27] N. Gandhi, E. Roth, B. Sandler, A. Haeberlen, and L. T. X. Phan. REBOUND: Defending distributed systems against attacks with bounded-time recovery. In *European Conference on Computer Systems (EuroSys)*, 2021.

[28] N. Gandhi, D. Saldaña, V. Kumar, and L. T. X. Phan. Self-reconfiguration in response to faults in modular aerial systems. *IEEE Robotics and Automation Letters*, 2020.

[29] N. Gandhi, J. Xu, D. Saldaña, and L. T. X. Phan. Online Rotor Fault Detection and Isolation for Vertical Takeoff and Landing Vehicles. In *Proc. IEEE Latin American Robotics Symposium (LARC)*, 2024.

[30] D. Gangadharan, L. T. X. Phan, S. Chakraborty, R. Zimmermann, and I. Lee. Video Quality Driven Buffer Sizing via Frame Drops. In *Proc. IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, 2011.

[31] R. Gifford, F. Galarza-Jimenez, L. T. X. Phan, and M. Zamani. Decntr: Optimizing Safety and Schedulability with Multi-Mode Control and Resource Allocation Co-Design. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2024.

[32] R. Gifford, N. Gandhi, L. T. X. Phan, and A. Haeberlen. DNA: Dynamic Resource Allocation for Soft Real-Time Multicore Systems. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2021.

[33] R. Gifford and L. T. X. Phan. Multi-mode on Multi-core: Making the best of both worlds with Omni. In *Proc. IEEE Real-Time Systems Symposium (RTSS)*, 2022.

[34] A. Haeberlen, L. T. X. Phan, and M. McGuire. Metaverse as a Service: Megascale Social 3D on the Cloud. In *Proc. ACM Symposium on Cloud Computing (SoCC)*, 2024.

[35] B. Kim, L. Feng, L. T. X. Phan, O. Sokolsky, and I. Lee. Platform-Dependent Timing Verification Framework in Model-Based Implementation. In *Proc. Design Automation and Test in Europe (DATE)*, 2015.

[36] B. Kim, L. T. X. Phan, O. Sokolsky, and I. Lee. A Model-Based I/O Interface Synthesis Framework for the Cross-Platform Software Model. In *Proc. International Symposium on Rapid System Prototyping (RSP)*, 2012.

[37] B. Kim, L. T. X. Phan, O. Sokolsky, and I. Lee. Platform-Dependent Code Generation for Embedded Real-Time Software. In *Proc. International Conference on Compilers Architecture and Synthesis for Embedded Systems (CASES)*, 2013.

[38] J. Lee, H. S. Chwa, L. T. X. Phan, I. Shin, and I. Lee. MC-ADAPT: Adaptive Task Dropping in Mixed- Criticality Scheduling. In *Proc. ACM International Conference on Embedded Software (EMSOFT)*, 2017.

[39] J. Lee, H. S. Chwa, L. T. X. Phan, I. Shin, and I. Lee. MC-ADAPT: Adaptive Task Dropping in Mixed- Criticality Scheduling. *ACM Transactions on Embedded Computing Systems (TECS)*, 16(5s):163:1–163:21, 2017.

[40] J. Lee, L. T. X. Phan, S. Chen, O. Sokolsky, and I. Lee. Improving Resource Utilization for Compositional Scheduling using DPRM Interfaces. In *Proc. Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS)*, 2010.

[41] J. Lee, S. Xi, S. Chen, L. T. X. Phan, C. Gill, I. Lee, C. Lu, and O. Sokolsky. Realizing Compositional Scheduling through Virtualization. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2012.

[42] H. Li, M. Xu, C. Li, C. Lu, C. Gill, L. T. X. Phan, I. Lee, and O. Sokolsky. Multi-Mode Virtualization for Soft Real-Time Systems. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2018.

[43] Y. Li, L. T. X. Phan, and B. T. Loo. Network Functions Virtualization with Soft Real-Time Guarantees. In *Proc. IEEE International Conference on Computer Communications (INFOCOM)*, 2016.

[44] J. Lian, N. Gandhi, Y. Wang, and L. T. X. Phan. Online Rotor Fault Detection and Isolation for Vertical Takeoff and Landing Vehicles. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024.

[45] C.-W. Lin, M. Di Natale, H. Zeng, L. T. X. Phan, and A. Sangiovanni-Vincentelli. Timing Analysis of Process Graphs with Finite Communication Buffers. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2013.

[46] Y. Litman, N. Gandhi, L. T. X. Phan, and D. Saldaña. Vision-based self-assembly for modular multirotor structures. *IEEE Robotics and Automation Letters*, 6(2):2202–2208, 2021.

[47] A. Loveless, R. Dreslinski, B. Kasikci, and L. T. X. Phan. IGOR: Accelerating Byzantine Fault Tolerance for Real-Time Systems with Eager Execution. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2021.

[48] A. Loveless, L. T. X. Phan, R. Dreslinski, and B. Kasikci. PCspooF: Compromising the Safety of Time-Triggered Ethernet. In *Proc. IEEE Symposium on Security and Privacy (S&P)*, 2023.

[49] A. Loveless, L. T. X. Phan, L. Erickson, R. Dreslinski, and B. Kasikci. CrossTalk: Making Low-Latency Fault Tolerance Cheap by Exploiting Redundant Networks. In *Proc. ACM SIGBED International Conference on Embedded Software (EMSOFT)*, 2023.

[50] H. Nguyen, R. Ivanov, L. T. X. Phan, O. Sokolsky, J. Weimer, and I. Lee. LogSafe: Secure and Scalable Data Logger for IoT Devices. In *Proc. IEEE International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2018.

[51] L. T. X. Phan, S. Chakraborty, and I. Lee. Timing Analysis of Mixed Time/Event-Triggered Multi-Mode Systems. In *Proc. IEEE Real-Time Systems Symposium (RTSS)*, 2009.

[52] L. T. X. Phan, S. Chakraborty, and P. S. Thiagarajan. A Multi-Mode Real-Time Calculus. In *Proc. IEEE Real-Time Systems Symposium (RTSS)*, 2008.

[53] L. T. X. Phan, S. Chakraborty, P. S. Thiagarajan, and L. Thiele. Composing Functional and State-based Performance Models for Analyzing Heterogeneous Real-Time Systems. In *Proc. IEEE Real-Time Systems Symposium (RTSS)*, 2007.

[54] L. T. X. Phan and I. Lee. Towards A Compositional Multi-Modal Framework for Adaptive Cyber-Physical Systems. In *Proc. International Workshop on Cyber-Physical Systems, Networks, and Applications (CPSNA)*, 2011.

[55] L. T. X. Phan and I. Lee. Improving Schedulability of Fixed-Priority Real-Time Systems using Shapers. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2013.

[56] L. T. X. Phan, I. Lee, and O. Sokolsky. Compositional Analysis of Multi-Mode Systems. In *Proc. Euromicro Conference on Real-Time Systems (ECRTS)*, 2010.

[57] L. T. X. Phan, I. Lee, and O. Sokolsky. A Formal Semantic Framework for Multi-Mode Systems. In *Proc. IEEE Real-Time Technology and Applications Symposium (RTAS)*, 2011.

[58] L. T. X. Phan, J. Lee, A. Easwaran, V. Ramaswamy, S. Chen, O. Sokolsky, and I. Lee. CARTS: A Tool for Compositional Analysis of Real-Time Systems. In *Proc. Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (CRTS)*, 2010. `http://rtg.cis.upenn.edu/carts/`.

[59] L. T. X. Phan, R. Schneider, S. Chakraborty, and I. Lee. Modeling Buffers with Data Refresh Semantics in Automotive Architectures. In *Proc. International Conference on Embedded Software (EMSOFT)*, 2010.

[60] L. T. X. Phan, M. Xu, J. Lee, I. Lee, and O. Sokolsky. Overhead-Aware Compositional Analysis of Real-Time Systems. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2013.

[61] J. Ren and L. T. Phan. Mixed-Criticality Scheduling on Multiprocessors using Task Grouping. In *Proc. Euromicro Conference on Real-Time Systems (ECRTS)*, 2015.

[62] R. Rothfeld, M. Fu, M. Balać, and C. Antoniou. Potential urban air mobility travel time savings: An exploratory analysis of Munich, Paris, and San Francisco. *Sustainability*, 2021.

[63] M. Sha, Y. Cai, S. Wang, L. T. X. Phan, F. Li, , and K.-L. Tan. Object-oriented Unified Encrypted Memory Management for Heterogeneous Memory Architectures. In *Proc. ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2024.

[64] D. Soudbakhsh, L. T. X. Phan, A. Annaswamy, and O. Sokolsky. Co-Design of Arbitrated Network Control Systems with Overrun Strategies. *IEEE Transactions on Control of Network Systems*, 5(1), 2018.

[65] D. Soudbakhsh, L. T. X. Phan, A. Annaswamy, O. Sokolsky, and I. Lee. Co-design of Control and Platform with Dropped Signals. In *Proc. ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2013.

[66] Y. Tang, N. Guan, W. Liu, L. T. X. Phan, and W. Yi. Revisiting GPC and AND Connector in Real-Time Calculus. In *Proc. IEEE Real-Time Systems Symposium (RTSS)*, 2017.

[67] Y. Wu, A. Chen, and L. T. X. Phan. Zeno: Diagnosing Performance Problems with Temporal Provenance. In *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2019.

[68] S. Xi, C. Li, C. Lu, C. D. Gill, M. Xu, L. T. Phan, I. Lee, and O. Sokolsky. RT-OpenStack: CPU Resource Management for Real-Time Cloud Computing. In *Proc. IEEE International Conference on Cloud Computing*, 2015.

[69] S. Xi, M. Xu, C. Lu, L. T. Phan, C. D. Gill, I. Lee, and O. Sokolsky. Real-Time Multi-Core Virtual Machine Scheduling in Xen. In *Proc. ACM International Conference on Embedded Software (EMSOFT)*, 2015.

[70] M. Xu, R. Gifford, and L. T. X. P. Phan. Holistic multi-resource allocation for real-time multicore virtualization. In *Proc. Design Automation Conference (DAC)*, 2019.

[71] M. Xu, L. T. Phan, H.-Y. Choi, Y. Lin, H. Li, C. Lu, and I. Lee. Holistic Resource Allocation for Multicore Real-Time Systems. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2019.

[72] M. Xu, L. T. Phan, O. Sokolsky, S. Xi, C. Lu, C. D. Gill, and I. Lee. Cache-Aware Compositional Analysis of Real-Time Multicore Virtualization Platforms. *Real-Time Systems Journal*, 2015.

[73] M. Xu, L. T. X. Phan, H.-Y. Choi, and I. Lee. Analysis and Implementation of Global Preemptive Fixed-Priority Scheduling with Dynamic Cache Allocation. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2016.

[74] M. Xu, L. T. X. Phan, H.-Y. Choi, and I. Lee. vCAT: Dynamic Cache Management using CAT Virtualization. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2017.

[75] M. Xu, L. T. X. Phan, I. Lee, O. Sokolsky, S. Xi, C. Lu, and C. D. Gill. Cache-Aware Compositional Analysis of Real-Time Multicore Virtualization Platforms. In *Proc. IEEE Real-Time Systems Symposium (RTSS)*, 2013.

[76] T. Yang, R. Gifford, A. Haeberlen, and L. T. X. Phan. The synchronous data center. In *17th Workshop on Hot Topics in Operating Systems (HotOS '19)*, May 2019.