Research statement

My research vision is to bring the *ultra-reliability* of today's commercial aircraft systems to the next generation of fully-autonomous Cyber-Physical Systems (CPS). My PhD thesis makes fundamental contributions towards realizing this vision in the context of *networked control systems*, which are key building blocks of modern CPS. In the future, I plan to focus on the design and analysis of ultra-reliable *networked autonomous systems with Artificial Intelligence (AI) components*, which will constitute the brain of next-generation CPS. My research methodology involves drawing from *real-time systems*, *distributed systems*, and *probability theory* foundations to design and analyze fault-tolerant systems with analytically quantified reliability guarantees.

1. Towards "ultra-reliable" cyber-physical systems

In a commercial aircraft, multiple fault tolerance mechanisms are in place to mitigate the effects of faults that may arise while airborne. In addition, reliability analyses are conducted to validate that the overall failure probability of the aircraft remains under a certified threshold even if faults occur at the maximum expected rates. To succinctly describe these practices, **researchers in the early seventies defined "ultra-reliability" as the practice of ensuring *quantifiably negligible* residual failure rates.**

Today, autonomous CPS are used in many domains, *e.g.*, autonomous vehicles, industrial automation systems, delivery and fire-safety drones, and telesurgery robots. If such autonomous CPS are to permeate our everyday life in the not so distant future, then they need to become at least as trustworthy as airplanes. However, the rigorous reliability engineering practices used in the avionics domain are expensive and cannot be replicated in most CPS domains, which run on small cost margins. For example, **using custom fault-tolerant hardware with triple-modular-redundant components and an additional set of spares is common in the avionics domain, but is not affordable for the automotive industry.**

Two recent trends further impede achieving aircraft-like reliability targets: the push towards the use of faster and cheaper, but less reliable, Commodity Off-The-Shelf (COTS) hardware, such as Ethernet; and the tremendous increase in the complexity of workloads used for next-generation CPS, *e.g.*, the use of Deep Neural Networks (DNNs) for self-driving cars. When such CPS are deployed in millions, for catastrophic consequences to occur, it suffices if just one of them experiences a faulty execution scenario that is different from those observed and studied during design time. Therefore, new and rigorous reliability analyses are necessary.

My goal is to **"enable" ultra-reliable *distributed real-time systems*** for the next generation of fully autonomous CPS, **despite the cost and complexity challenges** mentioned above.

2. PhD Thesis: Reliability analysis of networked control systems

A Networked Control System (NCS) comprises one or more control loops wherein the control and feedback **signals are exchanged among *distributed* components** through a communication network. To ensure a minimum quality of control, the **underlying infrastructure must provide strong temporal guarantees** (also referred to as *hard real-time* guarantees). However, implementing ultra-reliable distributed real-time systems in a cost-effective manner using COTS processors and networks for deploying safety-critical NCS applications is far from trivial. One aspect that makes the problem particularly difficult is the effect of the harsh environments in which CPS are often deployed. **Environmental disturbances cause *transient faults* (bit flips) in the hardware.** Although these transient faults occur with extremely low probabilities, they must nonetheless be tolerated to build trustworthy CPS. In the worst case, transient faults can cause host crashes, network retransmissions, and incorrect computations when an NCS is deployed over field buses such as Controller Area Network (CAN), and even *Byzantine errors* (*i.e.*, inconsistent broadcasts) in an Ethernet-based NCS.

In my thesis work, which is published at RTSS '15 [1], ECRTS '18 (Best Presentation Award) [2], ECRTS '19 [3], and RTAS '20 [4], I show how to analyze the reliability of NCS implementations in the presence of such transient faults. Unlike prior work, I analyze COTS-based hard real-time implementations of NCS applications to evaluate errors in both time and value domains, which is challenging because accounting for both vastly increases the number of system configurations that need to be analyzed for errors. Most importantly, **my thesis work is the first to formalize and propose techniques to eliminate *reliability anomalies* in a hard real-time setting.** Reliability anomalies can result in non-monotonic increases in a system's overall failure rate despite local decreases in an individual component's failure rate, resulting in *unsound* reliability estimates, *i.e.*, we might believe a system to be more safe than it actually is.

CAN-based systems. To illustrate the need for sound implementation-specific reliability analyses more clearly, consider a simple scenario where an NCS is deployed over CAN. The CAN protocol detects network corruptions due to transient faults using checksums, and automatically retransmits any corrupted messages.

However, retransmissions do not protect against node crashes, or against memory corruptions before the checksum is computed. Thus, a common approach to tolerate such node errors, without affecting the NCS's quality of control, is to *actively replicate* software tasks (responsible for sensing and control) on separate processors.

Intuitively, retransmissions and active replication are together expected to make the system more resilient to fault-induced errors. On the contrary, **we showed that adding replicas beyond a certain point actually hinders a system's ability to meet all deadlines, and can ultimately decrease its overall reliability [1].** For a systems reliability engineer, this poses tough design questions. Clearly, replicating the most-critical tasks to some extent is a good idea, but **what is the "right" number of replicas?** Should one choose triple modular redundancy, or would two replicas suffice to achieve the desired reliability? Perhaps five replicas would be much better? Or would this already be detrimental from a timing-reliability point of view?

To answer such questions while tackling the search space explosion caused by the consideration of both time and value domain errors, **I proposed a multi-layered approach for reliability analysis of an actively replicated NCS that is implemented over CAN [2].** My analysis (i) models transient faults using a Poisson distribution; (ii) upper-bounds the probability of fault-induced *basic errors* such as node crashes, memory corruption, and network message corruption; (iii) upper-bounds the probability of *application-specific message errors* such as message omissions, incorrect computations, and retransmissions; and finally (iv) **upper-bounds the probability that the final actuation during a single NCS iteration deviates from an error-free execution.** I make use of real-time scheduling theory to obtain accurate bounds on the lifetime of tasks and messages, and probability theory to obtain safe bounds on the expected NCS failure rate.

Analysis of Ethernet-based systems. In the future, Ethernet-based NCS are expected to play a more important role due to Ethernet's low cost and high bandwidth. However, Ethernet is fundamentally different from field buses like CAN. **Lack of an atomic broadcast primitive exposes Ethernet-based systems to the risk of environmentally-induced Byzantine errors. Byzantine fault tolerance (BFT) protocols can mitigate such errors to a large extent, but is the cost of using a BFT protocol offset by the gain in reliability?** Is one network topology necessarily or significantly more reliable than another? Prior work has not analyzed BFT protocols in the context of distributed real-time systems [5], nor has it accounted for the network topology and the non-uniform fault rates across different network components that arise due to environmental disturbances. Answering such questions therefore requires new kinds of analyses.

To address this gap, **I proposed the first quantitative reliability analysis of a hard real-time BFT atomic broadcast protocol over Ethernet in the presence of stochastic transient faults [4, 6].** Once an actively replicated NCS is implemented over an atomic broadcast layer, it has similar failure semantics as that of an implementation over CAN. Thus, composing the result of this analysis with the result of the CAN analysis above [2] enabled analyzing the entire software stack (active replication, atomic broadcast, and Ethernet's transmission protocol). Specifically, I explored different implementations of actively replicated NCS over commodity Ethernet to reveal non-trivial reliability trade-offs, such as the significant impact of the number of message exchange rounds and the network topology on the overall reliability [4].

Temporal robustness. The above analyses provide an upper bound on the failure probability of a *single* NCS iteration. However, such per-iteration results are often insufficient for answering whole-system reliability questions in the context of system operation times. For example, given a fleet of one million autonomous cars, each operating for five hours a day on an average, **if it is desired that less than ten cars break down due to transient faults in a year, a reliability metric like the *Failures-In-Time (FIT)*—*i.e.*, the expected number of failures in one billion operating hours—is much more useful.**

The conventional approach to obtain metrics like FIT from single-iteration estimates is to calculate the time to first fault. However, this approach is overly simplistic and grossly pessimistic for NCS applications, which are routinely designed to be *temporally robust*, *i.e.*, to remain functional despite a few skipped or misbehaving control-loop iterations. Instead, **I proposed analyses that account for the temporal robustness of an NCS using stateful models such as the widely used *weakly-hard constraints*, resulting in vastly more accurate FIT estimates [3, 7].** In particular, I presented three different techniques based on *probabilistic model checking*, *martingale theory*, and *sound approximation* to address the expressiveness, accuracy, and scalability requirements of larger, more complicated, and diverse NCS applications.

3. Other projects: Scheduling for real-time and datacenter systems

In addition to my research on the reliability analysis of NCS and distributed real-time systems, I have also contributed to the areas of hard real-time scheduling theory and cloud computing. These contributions have been published at **ECRTS '13 (Outstanding Paper Award) [8], RTSS '14 [9], RTS Journal '15 [10], Middleware '17 (Best Student Paper Award) [11], and EuroSys '18 [12].**

Schedulability analysis of Linux-like systems. In a real-time operating system, offline validation (typically known as *schedulability analysis*) is used to analyze whether a scheduling policy and its runtime implementation

is guaranteed to always satisfy the timing requirements of all hard real-time tasks in the system. The large body of literature on multiprocessor schedulability analysis focused mainly on *partitioned* scheduling (each task is scheduled only on a specified core) and *global* scheduling (all tasks can be scheduled on all cores), and to some extent on *clustered* scheduling (global scheduling on disjoint clusters of cores).

In practice, contemporary real-time operating systems, such as VxWorks, LynxOS, QNX, and real-time variants of Linux, allow tasks to have *arbitrary processor affinities* (APAs), which is analogous to clustered scheduling with possibly overlapping clusters. To enable the use of APAs in safety-critical CPS, **I presented the first techniques for the schedulability analysis of real-time tasks with APAs [8, 10], and a novel *strong APA scheduling* policy for better utilization of the CPU [9].** This work subsequently spawned several research projects on APAs. Peter Zijlstra (co-maintainer of the Linux kernel scheduler) in his keynote at ECRTS 2017 highlighted our community's efforts to bridge this specific gap between theory and practice.

Scheduling in the cloud. During my internship at Microsoft Research, I worked on the problem of *autoscaling*, *i.e.*, automatically spawning and shutting down frontends (which receive and schedule client requests) and backends (which compute responses) based on their workload. The production team had reported poor resource efficiency in their infrastructure, which consisted of hundreds of frontends and backends, for serving machine learning inference requests. I designed a predictable load balancer and request scheduling mechanism, and an analytical model for estimating the resulting queuing latencies at runtime. Based on the proposed design and its analytical model, I then **developed a *distributed autoscaler* called *Swayam* [11], which could *proactively* autoscale the number of frontends and backends to satisfy the dual objectives of guaranteeing low end-to-end latencies and minimizing resource usage. Microsoft Azure's Swayam-based framework was deployed in 2016 and has hosted over 100,000 services to date.**

More recently, I worked on the problem of *virtual machine (VM) scheduling* for the cloud environment. Contemporary VM schedulers leave substantial room for improvements. For example, when confronted with a large number of VMs, the widely used *Credit* scheduler in Xen can negatively affect tail latencies and throughput due to the use of unpredictable scheduling heuristics, or due to implementation aspects that cause undesirably high overheads. **We proposed a novel scheduler *Tableau* [12] inspired by hard real-time systems to minimize runtime overheads while maintaining high throughput and predictable latencies, even with a high VM density. Ideas from Tableau are currently being used in Google's cloud scheduler.**

4. Future directions

The landscape of autonomous safety-critical CPS is rapidly changing. There is a strong push towards self-driving cars for improving road safety. There is also an increase in privatization in the space sector, access to robot development tools (*e.g.*, Amazon's RoboMaker platform), and the use of drones for a variety of purposes. Motivated by these trends, I will pursue two main research thrusts: **(i) Reducing the cost of making intelligent networked autonomous systems as reliable as an actively replicated NCS with triple modular redundancy. (ii) Pushing the envelope of existing reliability analyses further to encompass implementations on modern hardware, and by using white-box analysis approaches.**

DNN frameworks for time-sensitive workflows. Control loops consisting of DNNs will be responsible for making critical scene recognition and trajectory planning decisions on the fly in the next generation of fully autonomous CPS. However, **certifying DNN implementations is quite challenging.** Unlike conventional software, DNNs are guided by millions of tunable parameters, which means analyzing their entire state space is not feasible. Furthermore, to meet strict end-to-end latency goals (*e.g.*, 100 ms in autonomous vehicles), DNNs are typically executed on highly parallel accelerator platforms, such as Google's TensorFlow and MIT's Eyeriss, which have not yet been analyzed from a safety or timeliness perspective.

Despite these challenges, I believe networked autonomous systems with DNN components can be valuable assets in developing rich, novel CPS applications. Therefore, to enable safety-critical CPS benefit from such intelligent autonomous systems, I am interested in building ultra-reliable DNN frameworks. In particular, recent studies have shown that DNNs have intrinsic resilience against transient faults owing to their sparse network structure, but the resilience of different structural units (*e.g.*, layers and neurons) within a single DNN differ significantly. While this calls for a systematic reliability analysis of DNN frameworks, there is also an **opportunity to design DNN frameworks that are as resilient as a triple-modular-redundant system (*i.e.*, with three functionally identical DNNs executing in parallel) but at roughly one-third the cost. Conceptually, if the most critical computations inside a DNN can be accurately identified, we can selectively safeguard them through spatial, temporal, or algorithmic redundancy mechanisms.**

My goal is to design analyses to estimate a minimal spare *capacity* needed for achieving the desired reliability target (*e.g.*, the number of processing elements in a hardware accelerator to be reserved as spares) and propose mechanisms to efficiently orchestrate critical computations over this spare capacity while maximizing data reuse, so that end-to-end latencies remain under desired thresholds.

Resilient intelligence at the edge. Today's CPS need to perform complex resource-intensive computations on big data, which are typically run in the cloud. However, **latency, energy, bandwidth, and privacy concerns call for pushing more and more computation on the edge nodes**, which are closer to the data. For instance, an autonomous vehicle cannot rely on cloud-based computations for its essential functions, since the round trip time to the cloud is simply too high; and a fleet of drones operating at a remote location may not have enough energy and bandwidth to transmit all its data to the cloud for training and inference.

However, **executing learning-based workloads at the edge is challenging due to resource constraints of the edge nodes.** We can use specialized hardware to run machine learning algorithms if the size, weight, and power constraints of the CPS devices are not violated. Alternatively, we may need to trade off some learning accuracy in favor of resource efficiency. For example, in the case of DNN inference, we can use existing techniques to compress the model, or use staggered and hierarchical DNN architectures that allow for partial DNN execution. In some cases, it may be possible to distribute the workload across a cluster of edge nodes. **Finding a good solution among these hinges on whether the resource needs of the DNN framework as well as the resource constraints and performance of the edge nodes can be accurately characterized.** I am interested in facilitating such decision making using predictable designs that are rooted in real-time scheduling theory, and building a distributed framework for resilient intelligence at the edge.

Pushing the ultra-reliability envelope further. Synchronized clocks are essential for building safety-critical CPS (since certifying asynchronous designs is known to be much more difficult). Therefore, to compute the overall failure rate of a CPS relying on synchronized clocks, the clock synchronization module must be analyzed as an independent source of failure. However, the unpredictability in COTS-based implementations increases the minimum achievable clock skew, and renders existing provably correct clock synchronization algorithms to be only "intuitively correct." **I seek to bring COTS-based implementations of fault-tolerant clock synchronization algorithms into the ultra-reliability fold. Furthermore, I would like to use such quantifiably reliable clock synchronization algorithms to design more efficient distributed systems primitives for the cloud.** For example, if the probability with which the synchrony assumption is violated could be quantified in advance for datacenter workloads, we can replace asynchronous algorithms with synchronous algorithms, which can improve resource efficiency without violating response time SLAs.

Finally, to realize even higher resource savings, I am interested in improving the accuracy of existing reliability analyses using white-box approaches. In particular, **I intend to develop a finer-grained strategy—at the granularity of program variables—to more accurately upper-bound the probability of application-specific errors.** For example, using program analysis techniques, we can trace the propagation of bit flips from registers to program variables; identify program variables that, if corrupted, result in the payload corruption; and then upper-bound the probability of silent data corruption in the network message payload.

In summary, I am confident these directions, together, will go a long way towards enabling ultra-reliability in the next generation of autonomous, intelligent, and networked CPS.

References

[1] **Arpan Gujarati** and Björn B. Brandenburg. When Is CAN the Weakest Link? A Bound on Failures-in-Time in CAN-Based Real-Time Systems. In *RTSS 2015*.

[2] **Arpan Gujarati**, Mitra Nasri, and Björn B. Brandenburg. Quantifying the Resiliency of Fail-Operational Real-Time Networked Control Systems. **Best Presentation Award**. In *ECRTS 2018*.

[3] **Arpan Gujarati**, Mitra Nasri, Rupak Majumdar, and Björn B. Brandenburg. From Iteration to System Failure: Characterizing the FITness of Periodic Weakly-Hard Systems. In *ECRTS 2019*.

[4] **Arpan Gujarati**, Sergey Bozhko, and Björn B. Brandenburg. Real-Time Replica Consistency over Ethernet with Reliability Bounds. In *RTAS 2020 (to appear, preprint available at* https://www.mpi-sws.org/~arpanbg/temp/X8sbUKkJ93).

[5] Malte Appel, **Arpan Gujarati**, and Björn B. Brandenburg. A Byzantine Fault-Tolerant Key-Value Store for Safety-Critical Distributed Real-Time Systems. In *CERTS 2017 (co-located with RTSS 2017)*.

[6] **Arpan Gujarati**, Malte Appel, and Björn B. Brandenburg. Achal: Building highly reliable networked control systems. In *EMSOFT 2019 (work-in-progress track)*.

[7] **Arpan Gujarati**, Mitra Nasri, and Björn B. Brandenburg. Lower-Bounding the MTTF for Systems with (m,k) Constraints and IID Iteration Failure Probabilities. In *CERTS 2017 (co-located with RTSS 2017)*.

[8] **Arpan Gujarati**, Felipe Cerqueira, and Björn B. Brandenburg. Schedulability Analysis of the Linux Push and Pull Scheduler with Arbitrary Processor Affinities. **Outstanding Paper Award**. In *ECRTS 2013*.

[9] Felipe Cerqueira, **Arpan Gujarati**, and Björn B. Brandenburg. Linux's Processor Affinity API, Refined: Shifting Real-Time Tasks Towards Higher Schedulability. In *RTSS 2014*.

[10] **Arpan Gujarati**, Felipe Cerqueira, and Björn B. Brandenburg. Multiprocessor Real-Time Scheduling with Arbitrary Processor Affinities: From Practice to Theory. *Real-Time Systems*, 51(4):440–483, July 2015.

[11] **Arpan Gujarati**, Sameh Elnikety, Yuxiong He, Kathryn S. McKinley, and Björn B. Brandenburg. Swayam: Distributed Autoscaling to meet SLAs of Machine Learning Inference Services with Resource Efficiency. **Best Student Paper Award**. In *Middleware 2017*.

[12] Manohar Vanga, **Arpan Gujarati**, and Björn B. Brandenburg. Tableau: a high-throughput and predictable VM scheduler for high-density workloads. In *EuroSys 2018*.