Research Statement

Ang Chen

“The Internet is brittle and fragile and too easy to take down.” –Vint Cerf

My research goal is to make networks and distributed systems more robust. An increasingly large portion of our daily life depends on our networks, including e-commerce, video conferencing, and many other examples. Unfortunately, network outages are quite common, and they can cause significant damage – for instance, Google and Amazon can lose tens of thousands of dollars for every minute of downtime. These problems happen due to at least two reasons. First, networks and distributed systems are difficult to get right, as they are complex and large in scale. Even major data centers are not immune to network misconfigurations, software bugs, or ill-planned service updates. Second, they are plagued by various attacks, which are getting more advanced and harder to defend against. Recently, the U.S. East Coast experienced a massive outage due to a distributed denial-of-service attack by compromised IoT devices. To avoid these problems, it is necessary to build more robust networks and distributed systems than those that are available today.

My work addresses both of the above challenges. To diagnose non-malicious problems in networks and distributed systems, my thesis work takes an interdisciplinary approach that draws inspiration from the database community; it builds on, and substantially generalizes, the concept of data provenance to enable network diagnostics. To mitigate malicious attacks, I have designed techniques that can detect and diagnose timing anomalies that are caused by the attacker, and I have built systems that can guarantee a certain level of performance even when they are under attack.

My core areas are distributed systems and networking. I have collaborated with academic researchers in systems, databases, networking, and security, as well as industrial researchers and engineers from Akamai Technologies, Microsoft Research, and Huawei Technologies. I believe that collaborations, particularly those across (sub)disciplines, are a fruitful avenue to innovation. I find research topics by identifying pain points in real-world network operations – for instance, I have identified common themes of reported problems in major network management forums, and I have conducted interviews with network operators and engineers from companies such as Microsoft, Comcast, and Google. As a systems researcher, I validate my research ideas by building concrete software prototypes; for network data-plane applications, I also rely on hardware prototypes, e.g., on FPGAs.

Automated network diagnostics with provenance

In a complex network, many things can go wrong: for instance, nodes can be misconfigured, applications can interfere with each other, and software controllers can have bugs. In my thesis work, I have developed ways to diagnose problems in networks and distributed systems using data provenance. At a high level, provenance tracks causality between network states and events – for instance, it captures the fact that a packet was forwarded on a particular port because of a certain routing table entry. The tracked causality can then be used to comprehensively explain any event of interest, which provides a good starting point for network diagnostics. With my collaborators, I have designed and built systems to capture provenance on high-speed data [4, 7]; I have also developed several techniques that use the captured provenance to find the root causes of network problems [8, 9], to repair these problems [12, 13], and to avoid collateral damage when performing network repair [10].

Maintaining Internet-scale provenance [4, 7]: Using provenance in Internet-scale networks poses several challenges to existing provenance systems [7]. First, the Internet’s data plane has very high data rates, and capturing provenance at such rates can lead to substantial overheads. Second, collecting and querying provenance in a multi-domain network introduces privacy and security concerns. To address these challenges, I designed a system that provides secure packet-level provenance (SPP) [4] for the Internet’s data plane. In SPP, I designed a lightweight authentication scheme to provide security and privacy guarantees with low overheads, and I used hardware-software hybrid design to push expensive operations to FPGA hardware.

1
Our experimental results and case studies show that SPP can work efficiently at line rate, and that it can support a wide variety of diagnostic and forensic tasks.

Identifying root causes [8, 9]: In a large network, the provenance of a network event can contain hundreds of other states and events [9]. Therefore, pinpointing the root cause from the complex provenance information remains a significant challenge. However, many of the trickiest problems only affect part of the system, perhaps a subset of traffic or services. As a result, operators can typically identify “reference events” that are similar to the problematic event but are processed correctly—indeed, a packet was misrouted, but a similar packet was routed correctly. In this case, the root cause of the problem is likely to be whatever caused these packets to be treated differently by the network. Leveraging this insight, I developed a technique called differential provenance [8, 9], which can identify the root cause of a problem by reasoning about the differences between the problematic and reference events. Our experiments in the context of software-defined networks (SDN) and Hadoop MapReduce show that differential provenance can accurately pinpoint the misconfigurations in the system.

Generating network repairs [12, 13]: With my collaborators, I have also worked on generating network repairs using provenance. When applied to SDNs, classic provenance can easily reason about data—such as network configurations—but it assumes that the SDN controller program is immutable. However, in many cases, it is necessary to repair controller programs as well, since they can also be buggy. To address this, we developed a generalization called meta provenance [12, 13], which extends provenance to model both data and programs. Meta provenance works by treating programs as a special kind of data, and its provenance model incorporates both the program syntax and the operational semantics of the programming language. Our case studies in the context of SDNs show that meta provenance can generate effective repairs to controller programs within one minute.

Avoiding collateral damage [10]: Existing repair systems have a narrow interface that fixes individual problematic events, such as a misrouted packet. As a result, they can generate overly-specific repairs that cause collateral damage in the network. To address this problem, I designed NetGenie [10], which has a more expressive interface that allows operators to describe multiple constraints on what a correct network should be, and it uses causal networks—a generalization of provenance—to generate high-quality repairs.

Providing privacy guarantees [3]: When problems affect wide-area networks that span multiple domains, these domains can achieve more effective diagnostics if they allow each other to query their local data, such as packet-level provenance or statistics. However, this raises privacy concerns. In PRISM [3], I proposed an approach for domains to collaborate in diagnostics, while providing differential privacy guarantees.

Secure systems with reliable timing properties

In addition to handling benign faults, I have also worked on making networks and distributed systems more secure and reliable in the presence of attacks. In particular, my work focuses on defending against attacks that disrupt a system’s timing behavior—for instance, covert timing channels that manipulate the timing of network packets, or DDoS attacks that inflate the latency of requests. I have designed and built several secure systems with reliable timing properties even when they are under attack.

Detecting covert timing channels [5]: A covert timing channel is a way for an attacker to exfiltrate secret data by encoding it into the timing of network packets, instead of sending it directly in the open. Detecting timing channels is a longstanding problem in the security community, and many techniques have been developed over the years. However, existing detectors are all specific to particular kinds of channels with known timing patterns, and they do not work on new channels. I have developed a more general approach to channel detection using a novel technique called time-deterministic replay (TDR) [5]. TDR can reproduce the expected timing of a program’s execution with high accuracy, and, therefore, can be used to detect any timing modification to packets regardless of timing patterns. Our results show that TDR can detect a variety of state-of-the-art channels as well as new channels, without any false positives or negatives.

Recovering from attacks within bounded time [11]: Cyber-physical systems (CPS) are a special kind of distributed system that interacts with the physical world. To defend against attacks on CPS, traditional fault-tolerance techniques are not enough: a) they tend to sacrifice liveness to preserve safety, which is insufficient because they cannot provide crucial timing guarantees in CPS, and b) they tend to mask every single fault by replication, which is costly in CPS, where resources are scarce. My collaborators and I proposed a new technique called Bounded-time Recovery (BTR) [11]. BTR leverages a special property of CPS—their physical components often have inertia or thermal capacity, and their control algorithms are already resilient to noisy inputs, so short malfunctions typically do not cause damage. Therefore, BTR does not attempt to mask all faults—it allows short faults to happen, but guarantees that they are detected and
recovered from within a bounded amount of time. By doing so, it can provide strong security guarantees with a lower overhead.

**Providing performance guarantees under DDoS attacks** [6]: In SplitStack [6], my collaborators and I proposed a novel DDoS defense that can provide better performance guarantees than existing approaches. SplitStack has a new software architecture, where a monolithic software stack is split into many small functional units; and it has a centralized controller, which schedules these units to run on available machines. When under attack, the SplitStack controller replicates impacted units across the data center to achieve desired timing and throughput guarantees. Our initial results suggest that SplitStack can provide better performance guarantees when under attack, and we are actively developing SplitStack in our ongoing work.

**Network measurement**

In order to make networks and distributed systems more robust, it is important to gain a deeper understanding of their current status. To this end, I have designed and built tools to conduct large-scale measurement studies [1, 14, 2, 15]. Measuring a network path’s capacity is an important capability in many applications, such as video streaming and online games. In [1], my collaborators and I showed that existing capacity measurement tools cannot obtain accurate results when a path’s capacity is highly asymmetric in forward and reverse directions. We designed a new measurement tool called TRIO, which is robust to capacity asymmetry. Our mathematical modeling and live Internet measurements show that TRIO can conduct accurate measurements in scenarios where other tools fail.

Traceroute data analytics is another useful way of characterizing path properties, including path diversity, routing changes, and path asymmetry. In [2], I showed that existing tools for this task perform redundant operations, therefore they are inefficient on big datasets. To address this, I designed a new algorithm called rtd to eliminate this redundancy and speed up analytics on big traceroute data.

I have also used measurement studies to gain insights into better system design. With my collaborators, I performed a measurement study on Akamai’s NetSession system [14], which is a commercial content distribution network with more than 24 million users. This study identified several interesting benefits and risks in Akamai’s hybrid architecture, which allows users to download from both the Akamai infrastructure and other peering users. In another work [15], we used measurement results from social networks to design onion routing protocols that can provide better privacy.

**Future work**

I plan to build on my expertise in designing and building robust distributed systems, and to extend my research agenda to the following directions.

**Quantitative diagnostics in distributed systems:** I plan to extend my work on provenance to diagnose quantitative problems in distributed systems, including probabilistic events, such as random packet drops, and aggregation properties, such as low throughput or high latency. This involves a new set of research challenges. First, classic provenance only captures functional causality, but not timing causality; therefore, it can only explain why an event happened, but not why it happened at a particular time. To achieve the latter, it may be necessary to enhance the provenance model to capture timing causality. Second, classic provenance has a deterministic model, so it does not work well on probabilistic failures, such as random packet drops. In those scenarios, erroneous events may still be considered “non-problematic” as long as their aggregated amount is under a certain threshold. Addressing this challenge may require developing a probabilistic provenance model. Third, random failures may lead to incomplete provenance information. Therefore, it is useful to develop provenance systems that can offer probabilistic explanations even in the presence of partial data loss.

**Fully-automatic fault recovery:** I also plan to explore the possibility of taking human operators “out of the loop” in fault recovery. I envision a kind of reliable distributed system where faults can be detected and corrected without humans getting involved. In such a system, a fault detector will detect any problems automatically, and report them to a fault correction component. The fault correction component will apply a network repair, and instruct the fault detector to continue monitoring the network conditions. This cycle repeats until a set of recovery criteria is met. Achieving this goal will likely involve addressing research challenges in several areas, including formal methods (when specifying correctness), runtime verification (when detecting faults), or even real-time scheduling (when performing timely recovery).
Leveraging emerging hardware platforms for security: I would also like to identify more opportunities to leverage emerging hardware platforms to build secure distributed systems. First, secure distributed systems typically involve heavy cryptographic operations. Therefore, they can potentially benefit from hardware acceleration, or hardware-software co-design, where expensive operations are performed in hardware and lightweight postprocessing is performed in software. Second, dedicated hardware can also help with secure systems that need to process high-speed data. For instance, offloading certain DDoS detection and defense algorithms to FPGAs may enable a faster response when a data center is under attack. Third, distributed systems that require privacy protection may be able to leverage platforms such as Intel SGX— for instance, when multiple domains need to collaborate in diagnosis [3]. In the longer term, I also plan to enhance CPS security with specialized hardware— for instance, specialized hardware for cryptography may be needed in CPS, where computing resources are constrained; correctness of mission-critical tasks in embedded systems may benefit from trusted execution environments, such as Intel SGX or ARM TrustZone.

References


