

Network-Boosted Applications

Satadal “Sata” Sengupta

I am a **networked systems researcher** who builds systems that enable networks to actively boost Internet applications. Modern Internet applications such as live streaming, video conferencing, AI chatbots, and emerging augmented/virtual reality systems are pushing networks to their limits, demanding far more in performance, security, and scale than the Internet’s best-effort design was built to provide. Yet many of the mechanisms used to manage performance and defend against attacks still rely primarily on end hosts or servers, even when the underlying problems emerge inside the network itself. The consequences are increasingly visible: short-lived congestion can degrade video quality, routing attacks can silently divert traffic and steal sensitive documents, and inefficient implementations can make widely used applications prohibitively expensive to operate at scale. My research addresses this gap by building systems on **programmable network platforms** that measure application-level signals from live traffic and use them to drive **real-time in-network intervention** to improve performance, security, and scalability.

This vision, which I call **network-boosted applications**, unfolds in layers. The network must first recover application-level signals from packet streams, then measure them continuously at scale, use them to drive real-time intervention, and, when beneficial, directly execute selected application functions. Across these four layers, my work spans multiple studies. Below, I highlight one flagship example from each.

- **Demystifying opaque applications from the network’s vantage point.** In joint work on **Zoom**, I showed that even a proprietary, encrypted real-time application leaves enough information on the wire to recover quality-of-experience metrics such as bitrate, frame rate, latency, and jitter. These methods **enabled the first large-scale network-centric analysis** of Zoom performance [1].
- **Enabling deep observability at production scale.** I designed **DART**, a system to continuously and accurately measure round-trip time (RTT) from live TCP traffic on terabit-scale programmable switches [2]. DART retains the minimal per-flow and per-packet state needed to resolve ambiguities in RTT computation within tight switch memory constraints. On campus traces, its prototype **recovered >99% RTT samples** collected by an offline unlimited-memory software baseline.
- **Driving real-time in-network intervention.** I developed **HiDe**, a system that detects long-distance routing attacks in real time using data-plane measurements [3]. Because such attacks induce long detours, sharply increasing RTT, HiDe monitors RTT and applies lightweight change detection that scales to production traffic and separates attacks from benign events. HiDe’s switch prototype **detected hijacks within 500 ms** with a **false-positive rate below 0.012%**.
- **Offloading application functionality into network hardware.** I co-designed **Scallop**, a novel hardware-software architecture that offloads the high-volume, latency-sensitive tasks of modern video-conferencing servers into programmable switches and smart Network Interface Cards (SmartNICs), while retaining complex control tasks in software [4]. Scallop’s switch prototype **handled 99% of bytes in hardware** and **scaled 10–200×** beyond a 32-core commodity server at the same cost.

My research is **informed by operational insights** from Princeton’s Office of Information Technology, ongoing technical engagement with NVIDIA, and three research internships at Netflix. It has **produced open-source tools and hardware prototypes** on programmable network platforms, with measurement and evaluation **grounded in anonymized production traces** from Princeton’s campus network. This work has been featured twice on the APNIC community blog, used in networking courses including at TU Delft, and has informed follow-on research in academia and industry. These results show how networks can boost applications. As applications evolve, my future research aims to develop **principled approaches for networks and applications to co-evolve**, enabling gains that neither can achieve alone.

Demystifying Opaque Applications in the Network

Many important Internet applications remain operational black boxes to the networks that carry them because they rely on proprietary protocols and widespread encryption. This opacity makes it difficult for network operators and application providers to diagnose problems, improve user experience, and reason

about security and policy, even when the network is best positioned to do so. Yet applications often still leave behind small but useful pieces of unencrypted information in packets, either because the protocol requires them for operation or because the implementation leaks them unintentionally. My work in this area asks what the network can still learn from this information by identifying the minimal signals left on the wire and determining what application-level behavior can be recovered from them.

- **Zoom [1].** In joint work on Zoom, we show that a proprietary, mostly encrypted application still exposes enough structure for the network to recover user-facing performance metrics. We begin with an entropy analysis of packet bytes to identify unencrypted fields, then use controlled experiments to infer their semantics. This let us reconstruct Zoom’s media hierarchy from packets to frames to streams to meetings, while distinguishing audio, video, and screen sharing. Using this structure, we recover bitrate, frame rate, jitter, and end-to-end latency from packet traces. Applied to month-long campus data, our methods yielded the first large-scale network-centric study of Zoom performance and revealed a long tail of poor quality across key indicators. I contributed broadly to this study and led development of the measurement infrastructure used to collect and analyze campus traffic.
- **Earlier work.** This line of work began before my Ph.D. and spans several other opaque systems. In joint work on YouTube, we showed through browser instrumentation and controlled experiments that the client adapts not only bitrate but also segment length, a previously undocumented behavior that helped explain why earlier studies misjudged its efficiency [5]. In work on TLS-encrypted mobile traffic, I showed that diversity in TLS configurations and cipher suites leaves signatures that improve application classification accuracy from 62% to 95% [6]. In joint work on Free Basics, we used crowdsourced measurements across 15 countries to show both the reach and limitations of this popular free service; many popular sites were excluded, free-service pages were substantially slower than paid alternatives, and much of the linked content fell outside the free ecosystem [7].
- **AI chatbots (ongoing work).** I am extending my work to AI chatbots like ChatGPT, which are rapidly becoming a major Internet workload. In an advisory role, I am helping guide a measurement study that uses controlled experiments and campus data to characterize their network footprint.

Enabling Continuous In-Network Monitoring

Recovering application-level signals from packet streams is only the first step. To be useful in practice, the network must also measure them continuously from live traffic, at production scale. Existing approaches are limited: active probes add overhead, end-host instrumentation is difficult to deploy, and coarse summaries often miss the short-lived events that matter most for performance and security. Programmable switches offer the best opportunity because they sit directly in the data path and operate at terabit speeds. Yet they do so under severe constraints: packet processing follows a fixed pipeline, each stage supports only simple operations, and on-chip memory for per-flow state is scarce. My work in this area asks whether the network can still maintain an always-on view of application behavior by designing hardware-amenable mechanisms that continuously measure key signals within these constraints.

- **DART [2].** I designed DART (Data-plane Actionable RTTs), the first system to continuously and accurately measure TCP RTT from live traffic on terabit-scale programmable switches. RTT is a key signal for diagnosing application slowdowns, detecting routing anomalies, and enabling timely rate control, but measuring it continuously in the data plane is challenging. TCP acknowledgments provide a natural basis for RTT measurement: match a data packet with its returning acknowledgment and compare their timestamps. In practice, however, retransmissions and packet reordering make these matches ambiguous, while cumulative acknowledgments make stale packet records hard to identify and evict. These challenges are particularly acute on programmable switches, where memory is severely limited. DART overcomes them with protocol-aware correctness checks and a compact cuckoo-hash-based design that maintains only a small per-flow summary and a few packet records. Using simple sequence-number bounds, it rejects ambiguous matches and lazily removes stale records on hash collisions. On an Intel Tofino prototype, DART recovered 99% of the RTT samples produced by an offline software baseline based on tcptrace, while sustaining production speeds.
- **Continuous video-quality monitoring (ongoing work).** In ongoing work, I am using insights from our earlier Zoom study to build a continuous in-network video-quality monitoring system.

Driving Real-Time In-Network Control

Measurement is most valuable when it enables timely intervention. Many performance and security problems arise inside the network and unfold too quickly or too subtly for end hosts or offline analytics to address effectively. End hosts see only their own traffic and often react only after user quality-of-experience (QoE) or security has already been affected. Control-plane solutions can also miss stealthy problems or respond too slowly when every millisecond matters. My work in this area asks whether networks can use continuously measured *data-plane* signals to detect problems as they emerge and trigger immediate intervention, such as by dropping or rate-limiting traffic. The challenge is to do so accurately despite noisy traffic and under the tight resource constraints of programmable hardware.

- **HiDe [3].** I developed HiDe (Hijack Defense), a system that detects long-distance route interception attacks in real time using data-plane measurements from live traffic. These attacks are especially dangerous because they can reroute traffic along a long detour, exposing users' data while still delivering it to the destination and therefore remaining hard to notice. Existing defenses rely mainly on control-plane visibility, but sophisticated attacks can either evade visibility in the control plane altogether or become visible there only after enough time has passed to cause significant harm. HiDe instead detects attacks from within the network using RTT as a real-time signal of suspicious route changes. The key insight is that long geographic detours increase propagation delay, making them visible as sharp RTT increases in traffic. To separate attacks from benign delay variation caused by congestion and routine routing changes, HiDe tracks the minimum RTT per prefix as a denoised signal and applies a hardware-friendly changepoint detector that fits on programmable switches. Implemented on commodity switch hardware, HiDe detected ethically conducted hijacks within 500 ms while maintaining a false-positive rate below 0.012% on 12 hours of campus traffic.

Re-Architecting Networked Systems Across Hardware and Software

For increasingly demanding networked systems, measurement and control are sometimes not enough. A more powerful approach is to redesign the software-only system across hardware and software. The key idea is to identify which computations are high-volume, latency-sensitive, and naturally suited to packet processing, and run them on programmable hardware, while leaving richer, more stateful, or less frequent logic in software. My work asks how to decompose monolithic systems across programmable switches, SmartNICs, and servers while preserving functionality and improving performance and scalability.

- **Scallop [4].** In Scallop, I co-designed a hardware-software architecture for modern video conferencing. Scallop targets the selective forwarding unit (SFU), the server component that relays media among participants and adapts streams to each receiver's network conditions. As video-conferencing workloads grow, software SFUs can become a bottleneck, introducing jitter, degrading media quality, and making it increasingly costly to scale. Scallop starts from the observation that the most frequent, latency-sensitive SFU tasks—especially packet replication and selective forwarding—closely resemble traditional packet-processing operations. It therefore offloads these high-volume media operations to programmable switches and SmartNICs, while keeping infrequent and semantically richer control tasks such as session management and rate control in software. Achieving this split required overcoming several challenges, including efficient replication, per-receiver media adaptation, and compatibility with existing browser-based clients. Despite these challenges, Scallop's switch prototype handled 99% of bytes in hardware, supported up to 128,000 concurrent meetings on a single switch, and improved scaling by 10–200× over a 32-core commodity server at the same cost. I contributed broadly to Scallop and led the work on making application-layer replication feasible and efficient on programmable hardware while preserving compatibility with browser-based clients.
- **Intrusion detection systems (ongoing work).** In ongoing joint work, I am exploring how intrusion detection systems such as Zeek [8] can be decomposed across software and programmable hardware. These systems contain a high-volume packet-processing stage that generates events, followed by a higher-level engine that analyzes them for anomalies. We are investigating how to offload packet processing and event generation to programmable hardware while retaining event processing in software, including on SmartNIC CPU cores to free the host and improve scalability.

Future Research Agenda

I plan to pursue this agenda of network-boosted applications across three time horizons: short-term over the next 1–2 years, mid-term over 3–5 years, and long-term over 5–10 years and beyond. Across all three, I will build not only new systems, but also shared infrastructure that can help the field mature, including open-source tools, datasets, programmable-hardware prototypes, and realistic testbeds. I will draw on my industry engagements with Netflix and NVIDIA, my experience building and maintaining lab testbeds at Princeton, and my collaborations with campus network operators at Princeton to ground this work in realistic operational settings, constraints, and needs. These directions create strong opportunities for external funding and real-world impact at the intersection of networking, systems, security, and trustworthy AI. They will also help me continue catalyzing follow-on research, building on evidence of growing scholarly uptake, including usage of my open-source artifacts and nearly 500 citations.

Short-term: Extending Network-Boosted Applications to Emerging Workloads

In the short term, I will extend the lens of network-boosted applications to emerging domains such as AR/VR, multimodal AI assistants, and related latency-sensitive real-time systems. In these settings, low and stable latency is essential: excessive delay or jitter can break immersion, degrade responsiveness, and in AR/VR even make the wearer nauseous. Building on my prior and ongoing work on continuous monitoring and hardware-software co-design, I will focus first on tracking user-facing quality and early signs of degradation, and on prioritizing the most latency-sensitive media, control, and feedback traffic when the network is under stress. Because many of these workloads are media-heavy and may involve replication, I also want to explore whether their high-volume packet processing can be offloaded to switches or SmartNICs while richer session logic remains on servers. This direction will create opportunities for shared benchmarks, operator-facing tools, open-source artifacts, and industry-funded partnerships.

Mid-term: Safe and Trustworthy AI-Driven Network Control

Over the mid term, I want to move from improving individual workload classes to optimizing across many of them at once. As networks increasingly carry a mix of demanding applications, such as live streaming, immersive media, and multimodal AI assistants, prioritizing one class in isolation will not be enough. Operators will instead need fine-grained ways to steer the network toward the right QoE trade-offs across competing workloads under changing conditions. I therefore want to build an abstraction that gives operators meaningful QoE-oriented control knobs, and maps those knobs to measurable signals and coordinated actions throughout the network. AI is useful here because it can help explore a large control space and adapt to changing traffic mixes, but AI alone cannot ensure safety or earn operator trust: in network control, a single incorrect decision can have cascading consequences. My goal is therefore to pair AI-driven optimization with a conservative fallback layer on programmable platforms that continuously monitors network and application health, takes over when conditions become unsafe, and returns control to AI once the system stabilizes. This direction builds on my work on ML-based traffic classification [6, 9], ML-based QoE optimization [10], and real-time monitoring [2] and control [3] on programmable hardware. This also creates strong opportunities for NSF CISE/CNS funding, equivalent non-US funding, and industry funding, as well as for impact in operator-facing automation and trustworthy AI for infrastructure.

Long-term: Co-Designing Applications with the Network

In the long term, I want to move beyond optimizing applications and networks separately and enable them to be designed together. As QoE requirements grow, developers will need ways to couple application logic tightly to the underlying network substrate and to adapt quickly as that substrate changes in topology and capabilities. My goal is to build high-level abstractions that capture what parts of an application need high bandwidth, low latency, replication, state, or isolation, while hiding the low-level details of the heterogeneous platforms underneath. This will require new ways to program networking hardware and servers together, reason about their trade-offs in performance and flexibility, and compile those abstractions into efficient implementations. Because AI will increasingly help develop and maintain applications, these abstractions must natively support AI-assisted development. If successful, this agenda would help define a new model for networked systems and place me at the forefront of an emerging area where programmable networks become first-class participants in application design and evolution.

References

- [1] Oliver Michel, **Satadal Sengupta**, Hyojoon Kim, Ravi Netravali, and Jennifer Rexford. Enabling passive measurement of Zoom performance in production networks. In *Proceedings of the 22nd ACM Internet Measurement Conference (IMC)*, pages 244–260, 2022.
- [2] **Satadal Sengupta**, Hyojoon Kim, and Jennifer Rexford. Continuous in-network round-trip time monitoring. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 473–485, 2022.
- [3] **Satadal Sengupta**, Hyojoon Kim, Daniel Jubas, Maria Apostolaki, and Jennifer Rexford. Passive data-plane telemetry to mitigate long-distance bgp hijacks. In *1st New Ideas in Networked Systems (NINeS 2026)*, pages 14–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2026.
- [4] Oliver Michel, **Satadal Sengupta**, Hyojoon Kim, Ravi Netravali, and Jennifer Rexford. Scalable video conferencing using SDN principles. In *Proceedings of the ACM SIGCOMM 2025 Conference*, pages 1213–1231, 2025.
- [5] Abhijit Mondal, **Satadal Sengupta**, Bachu Rikith Reddy, MJV Koundinya, Chander Govindarajan, Pradipta De, Niloy Ganguly, and Sandip Chakraborty. Candid with Youtube: Adaptive streaming behavior and implications on data consumption. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 19–24, 2017.
- [6] **Satadal Sengupta**, Niloy Ganguly, Pradipta De, and Sandip Chakraborty. Exploiting diversity in Android TLS implementations for mobile app traffic classification. In *The World Wide Web Conference (The WebConf, formerly WWW)*, pages 1657–1668, 2019.
- [7] Rijurekha Sen, Hasnain Ali Pirzada, Amreesh Phokeer, Zaid Ahmed Farooq, **Satadal Sengupta**, David Choffnes, and Krishna P Gummadi. On the free bridge across the digital divide: Assessing the quality of Facebook’s Free Basics service. In *Proceedings of the 2016 Internet Measurement Conference (IMC)*, pages 127–133, 2016.
- [8] Zeek. <https://zeek.org/>. Accessed: 2025-11-01.
- [9] **Satadal Sengupta**, Vinay Kumar Yadav, Yash Saraf, Harshit Gupta, Niloy Ganguly, Sandip Chakraborty, and Pradipta De. MoViDiff: Enabling service differentiation for mobile video apps. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 537–543. IEEE, 2017.
- [10] **Satadal Sengupta**, Niloy Ganguly, Sandip Chakraborty, and Pradipta De. HotDASH: Hotspot aware adaptive video streaming using deep reinforcement learning. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, pages 165–175. IEEE, 2018.