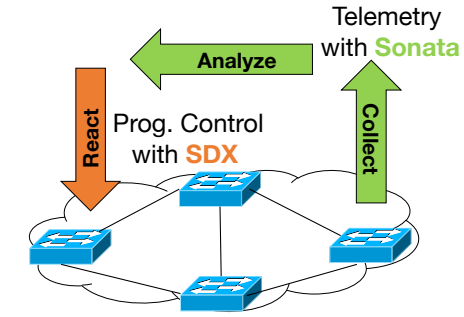
Research Statement

Arpit Gupta

As a systems researcher, I design and build easy-to-use, scalable, and deployable systems that solve real-world problems at the intersection of **networking**, **cybersecurity**, and **data science**. First, I identify real-world problems that network operators and vendors face while working "in the trenches" and understand their practical constraints. I then design innovative and provable solutions to these problems using various theoretical techniques (*e.g.*, optimization, graph theory, machine learning) and emerging technologies (*e.g.*, programmable data plane, scalable stream processors). Finally, I build practical systems with minimal deployment overhead. My systems and the software packages I wrote are widely used in both academia and industry. For example, my software-defined Internet exchange point (IXP) prototype, which won the **Community Award** at *USENIX NSDI* in 2016, is used by many IXP operators across the globe. Also, the recently released, "network streaming telemetry" prototype, is currently being used by network operators and researchers at AT&T.

How can networks run themselves? My dissertation research centers around answering this question. In recent years, with the proliferation of networked devices (e.g., Internet-of-Things), systems (e.g., container networks), and applications (e.g., augmented or virtual reality), the complexity of managing networks at scale has increased significantly. Conventional network management tools and practices built for much simpler networks are ill-suited to handle this increased complexity. To satisfy the increased security, availability, and performance requirements of modern networks; my goal is to design and build **self-driving networks** that automatically (without humans in the loop) make holistic (not protocol-specific) control decisions in real time.

To mitigate the effects of dynamic uncertainty, self-driving networks need to "sense" (*i.e.*, collect **high-velocity** network data), "infer" (*i.e.*, analyze collected **high-volume** data), and "actuate" (*i.e.*, react to different inferred network events) by themselves; that is, they rely critically on closed-loop feedback. My dissertation research focuses on the expressiveness, scalability, and correctness of two key building blocks for this closed feedback loop:



Feedback loop for self-driving networks.

- *Telemetry* [2,4,7], which refers to the process of collecting and analyzing the raw network data to discover various network activities of interest (*e.g.*, inferring presence of DDoS attacks and link failures) in real time; and
- *Programmatic Control* [1,3,5,6,9,10], which refers to the process of programmatically applying the fine-grained reactive actions without disrupting networks' default behavior (*e.g.*, redirecting traffic from congested peering links without creating any forwarding loops).

Telemetry with *Sonata*

Problem. For networks to run themselves, they need to monitor a wide range of network activities. For example, they need to concurrently detect whether the network is under attack and also determine whether there is a device failure in the network. This involves extracting multiple features from the traffic data and combining them to infer activities of interest in real time. Most existing real-time telemetry systems are either not expressive, *i.e.*, they support an insufficient set of telemetry tasks, or they are not scalable, *i.e.*, they fail to scale as the traffic volume intensifies and the number of telemetry tasks increases. In contrast, self-driving networks require running *multiple expressive* telemetry queries over *high-volume* networks. My research focuses on building a distributed real-time network telemetry system that is both *expressive* and *scalable*.

Observation. Many existing telemetry solutions take advantage of either scalable stream processing or programmable data-plane targets for network telemetry—but not both. While at first glance, these two technologies seem inherently different, they both apply a sequence of transformations over packets (tuples). This similarity provides new opportunities to combine their strengths. Also, since many telemetry tasks,

the fraction of relevant traffic is typically tiny. Thus, unlike existing task/query-agnostic systems, it makes sense to build query-driven telemetry systems. Based on these observations, I developed an expressive query-driven streaming telemetry system, *Sonata* [2, 4], that scales query execution by making use of both streaming and programmable data-plane targets and letting the output of queries themselves drive further processing.

Challenges and Solutions. Ideally, one would like to execute all queries in the data plane itself, but limited data-plane resources (*e.g.*, memory) restricts which portions of the queries to execute in the data plane. Requiring network operators to decide how to partition each query and configure the data-plane targets to run the partitioned queries by themselves can be overwhelming. To address this problem, Sonata provides a declarative query interface where network operators express queries for a range of telemetry tasks as a sequence of dataflow operations over packet tuples. Internally, Sonata takes these queries, representative workload data, and various data-plane constraints and determines the query partitioning plan that minimizes the workload at the stream processor. Sonata's query planner also determines a query-specific refinement plan that iteratively zooms-in over portions of traffic that satisfy the queries, making the best use of limited data-plane resources.

As a first step, I focused on a single-switch implementation for Sonata [2, 4]; I'm currently extending this work to build a network-wide telemetry system. For the network-wide settings, Sonata's query planner needs to determine, among other issues, where to evaluate each query; how to partition the stateful operations over multiple hops while making the best use of limited resources along a path; and how to select the thresholds for filter operations [7].

Impact. Building an expressive and scalable telemetry system is an important step towards making self-driving networks a reality. This project has generated significant interest in self-driving networks, with two recently announced workshops (NSF and ACM SIGCOMM) focusing exclusively on this topic. I have also open-sourced this project, and I am currently working with researchers at ISPs (AT&T) and security vendors (NIKSUN Inc.) to express and scale queries for various real-world telemetry tasks with Sonata.

Programmatic Control with *SDX*

Problem. After determining the reactive actions with telemetry, a self-driving network needs to apply these fine-grained actions in the data plane, usually in multi-domain settings. My research focuses on applying programmatic control for the inter-domain networks settings, where by default networks use the Border Gateway Protocol (BGP) to exchange traffic with each other. In my study, I demonstrate that BGP is not suited for applying fine-grained control actions. Ideally, one would like to replace BGP with a clean-slate solution supporting programmatic control at scale. However, the existing Internet-wide deployment of BGP-speaking routers makes such an approach impractical. Thus, the goal of my research was to design and implement a dirty-slate solution that ensures maximal impact with minimal deployment overhead while safely interoperating with BGP.

Observation. In recent years, we have seen the emergence of Internet exchange points (IXPs). They provide a common switching fabric for various networks for exchanging traffic with each other and are strategically located to influence a significant portion of the Internet's inter-domain traffic. I proposed building a software-defined exchange (SDX) [6] that replaces the conventional switching fabric with programmable switches at IXPs. While simple in theory, my research shows that building a practically deployable SDX requires striking a delicate balance between expressiveness, correctness, privacy, and scalability.

Challenges and Solutions. Though SDX enables flexibility at the IXP, everyone else is still using BGP. Thus, it needs to ensure that this additional flexibility does not come at the cost of creating forwarding loops in the network. Requiring SDX participants to express their control program while guaranteeing forwarding correctness can be cumbersome, as they not only have to worry about their own control program but also their peers' programs. To address this problem, SDX provides a virtual switch abstraction to each participant which they can use to express their policies without worrying about others. SDX augments participant's programs to ensure correctness. It then composes them together and compiles the composition to forwarding table entries for the data plane. While such an augmentation ensures correctness, the cost of additional forwarding rules in the data plane can create scalability problems. I developed novel "attribute encoding" schemes [5, 6] that make use of several existing protocols and mechanisms to encode the for-

warding attributes in the header of each packet before it enters the switching fabric—reducing the number of forwarding table entries required in the data plane.

To ensure that SDX participants can trust the computations at the IXP, I designed and implemented *SGDX* [8], that uses trusted execution environments such as Intel SGX to keep participants' policies private. To ensure that SDX participants can only influence the traffic that they are authorized to over shared physical switch, I built *FLANC* [3], which authorizes action requests over a shared switch for each participant. Finally, to stop participants at multiple IXPs from expressing conflicting policies, I developed *SIDR* [1], which makes it easier for participants to trade expressiveness (*i.e.*, fine-grained control policies) with privacy (*i.e.*, share control policies with other SDXs) to ensure correctness.

Impact. My SDX paper [6] had a significant impact on the research community. It was the first project to demonstrate how programmable switches can enable flexible inter-domain routing at scale. It inspired NSF's first workshop on SDX in 2014, which was followed by multimillion-dollar grants for research in this area. The work also inspired the creation of Endeavour, a multi-university consortium in Europe for SDX-related research. To date, my SDX paper [6] has over 210 citations making it one of the **highest cited** papers from *SIGCOMM 2014*.

I open-sourced the *iSDX* [5] project with the Open Networking Foundation (ONF) and worked closely with two full-time software developers to make the *iSDX* code production ready. I also worked with various network operators, switch vendors, and service providers to help deploy SDX at an inter-government agency exchange and many IXPs across the globe. Project *Endeavour* currently uses *iSDX* as the default platform for their projects. For these efforts, the project won the **Community Award** at *USENIX NSDI 2016*. Also, our *PathSets* [10] paper, that demonstrates how iSDX's encoding scheme are applicable for more general network settings, won the **Best Paper** award at *SOSR 2017*.

Future Directions

Sonata and *SDX* are only two basic building blocks for self-driving networks. In the future, I would like to build upon these efforts to make the distant dream of self-driving networks a reality.

Co-designing Machine Learning and Query Planning Algorithms

Self-driving networks need to train learning models to discover activities of interests. Currently, learning algorithms only focus on training the learning models to maximize accuracy, and query planning algorithms solely concentrate on minimizing the execution cost for queries that represent these learning models. This disconnect between the two often results in very accurate yet prohibitively expensive learning models that are not deployable in the production networks.

I observed that it is possible to augment the optimization problem that most state-of-the-art learning algorithms solve to trade accuracy against query execution cost while accounting various resource constraints (*e.g.*, limited memory in the data plane) to determine the learning model as well as the query plan. Based on this observation, I'd like to design and build a telemetry systems, *SonataML*, that co-designs the algorithms for the machine learning and query planning problems to train realistic and deployable learning models.

While the idea appears to be straightforward in principle, just embedding the cost and constraint metrics to the learning problems is not enough. Since the computational complexity of an optimal solution grows exponentially with the number of features for the learning problem, one needs to design heuristic algorithms that can find solutions to this problem that are just "good enough".

Another challenge is the availability of labeled training data. As most of the activities of interests are needles in a haystack, collecting labeled training data to bootstrap *SonataML* is extremely hard. I envision adapting existing active learning algorithms that can leverage the knowledge of the domain experts (security analysts and network operators) to generate as much of labeled data as possible while minimizing the workload for these human experts. As the nature of security and performance problems evolve with time, rather than removing humans from the loop altogether, I envision the active learning to be an intrinsic part of the self-driving networks, where the workload for the human experts reduces over time as the networks become more capable of running the networks by themselves.

Expanding Networking Telemetry Footprint

Self-driving networks require insights from multiple vantage points in the network. So far, *Sonata* focuses primarily on inferring network activities at the core locations of the Internet (*e.g.*, border routers for large ISPs). Although such deployments provide broader visibility with limited overhead, they miss several insights such as end-to-end performance for specific web applications (*e.g.*, Netflix, CNN, etc.) in a region. These insights, often available only at the edge of the Internet (*e.g.*, home routers, web browsers, etc.), are not only critical for robust self-driving networks but are also crucial to answering various policy questions on socio-political issues such as net neutrality, data privacy, censorship, etc.

I believe using cheaply available tools (e.g., programmable NICs and Raspberry Pis) to design a light-weight Sonata, deployed at home networks, can help expand the network telemetry footprint for self-driving networks. Here, in most cases, the entities generating data (producers) and processing data (consumers) are different. For example, quality-of-experience data for latency-sensitive applications are produced by end-users but can be consumed by an eyeball ISP to improve its network's performance. As a result, there is fundamental friction between the two entities, one trying to get as much data as possible and the other trying to preserve their privacy. Thus, the design of such a decentralized platform needs to execute queries in a privacy-preserving manner.

To ensure privacy-preserving query execution, I plan to design: (1) *Expressive User Interface* for end users to express *what* data to share and with *whom*. I plan to collaborate with programming languages and privacy experts to create a domain-specific language to express data sharing policies; (2) *Data Privacy and Integrity* guarantees for data producers and consumers respectively. To ensure data privacy, I plan to extend my previous work (*FLANC* [3]) to employ formal logic tools for generating policy compliance proofs for input telemetry queries. I also plan to expand, *SGDX* [8], to design protocols that make use of trusted execution environment at local nodes for guaranteeing data integrity; and finally, (3) *Model Privacy* guarantees for data consumers. Learning models are the intellectual property of the consumers, and they have good reasons to keep it private. To ensure model privacy, I plan to use secure multi-party computation to design protocols that distribute computations to local nodes such that privacy of learning models is preserved.

Beyond Self-Driving Networks

The computational complexity of machine learning models has grown significantly in recent years. To catch up with this increased computational requirements, most cloud service providers have responded by horizontally scaling the compute clusters. However, rather than simply throwing more resources at the problem, one needs to step back and think more carefully and explore if there are new technologies on the horizon that can make better use of limited available resources.

Programmable data-plane targets provide the opportunity to scale data analytics computations vertically. I believe *Sonata*'s idea of offloading query execution to data-plane targets applies to more general analytics settings. I plan to design and implement a **general-purpose analytics stack** that is not only aware of the compute capabilities and limitations of CPU-based compute nodes but also the set of data-plane targets connecting these nodes.

To make this system a reality, I need first to demonstrate that using data-plane targets for offloading analytics computations is cheaper than using more CPU nodes. Answer what kind of data center topologies will make the best use of programmable data-plane targets? What kind of targets to use, *i.e.*, whether one should deploy cheaper but low throughput targets (*e.g.*, smart NICs), or high throughput but expensive targets (*e.g.*, switching fabrics), or both? Also, whether one should use off-the-shelf programmable data-plane targets or design custom switch architectures tailored to data analytics tasks?

Second, I plan to design a query planner that for a given topology, compute nodes and programmable data-plane targets; determines packet processing pipelines that execute over both the data-plane and the streaming targets. The goal of this query planner is to minimize the execution time while considering various data-plane constraints (*e.g.*, limited memory) and limitations (*e.g.*, cannot perform multiplication or division operation). The query planner also needs to determine the data structures for packet's payload—reducing the parsing overhead for the intermediate data-plane targets. Finally, I'd also like to expand the idea of co-designing the algorithms for learning and query planning by applying it to more commonly-used machine learning problems.

References

[1] R. Birkner, A. Gupta, N. Feamster, and L. Vanbever. SDX-Based Flexibility or Internet Correctness?: Pick Two! In *ACM Symposium on SDN Research (SOSR)*, 2017. (Cited on pages 1 and 3.)

[2] A. Gupta, R. Birkner, M. Canini, N. Feamster, C. Mac-Stoker, and W. Willinger. Network Monitoring as a Streaming Analytics Problem. In *ACM Workshop on Hot Topics in Networks (HotNets)*, 2016. (Cited on pages 1 and 2.)

[3] A. Gupta, N. Feamster, and L. Vanbever. Authorizing Network Control at Software Defined Internet Exchange Points. In *ACM Symposium on SDN Research (SOSR)*, 2016. (Cited on pages 1, 3 and 4.)

[4] A. Gupta, R. Harrison, A. Pawar, M. Canini, N. Feamster, J. Rexford, and W. Willinger. Sonata: Query-Driven Network Telemetry. *Under Submission*, 2018. (Cited on pages 1 and 2.)

[5] A. Gupta, R. MacDavid, R. Birkner, M. Canini, N. Feamster, J. Rexford, and L. Vanbever. An Industrial-Scale Software Defined Internet Exchange Point. In *USENIX NSDI*, 2016. (Cited on pages 1, 2 and 3.)

[6] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett. SDX: A Software Defined Internet Exchange. In *ACM SIGCOMM*, 2014. (Cited on pages 1, 2 and 3.)

[7] R. Harrison, C. Qizhe, A. Gupta, and J. Rexford. Network-Wide Heavy Hitter Detection with Commodity Switches. *Under Submission*. (Cited on pages 1 and 2.)

[8] X. Hu, A. Gupta, A. Panda, N. Feamster, and S. Shenker. Preserving Privacy at IXPs. *Under Submission*. (Cited on pages 3 and 4.)

[9] H. Kim, J. Reich, A. Gupta, M. Shahbaz, N. Feamster, and R. Clark. Kinetic: Verifiable Dynamic Network Control. In *USENIX NSDI*, 2015. (Cited on page 1.)

[10] R. MacDavid, R. Birkner, O. Rottenstreich, A. Gupta, N. Feamster, and J. Rexford. Concise Encoding of Flow Attributes in SDN Switches. In *ACM Symposium on SDN Research (SOSR)*, 2017. (Cited on pages 1 and 3.)