Designing Provably Performant Networked Systems

Venkat Arun

My research is in the design, implementation, and analysis of networked systems. My work spans internet congestion control, video streaming, privacy-preserving computation, wireless networks, and mobile systems.

Across these areas, a unifying theme of my work is to bridge between heuristics that systems use in practice and proofs of how well they work. I have shown that current heuristics fail to provide performance guarantees, and I have used proofs to design more robust systems. My principal contribution is a set of tools and techniques to *prove performance properties* of real-world heuristics running in real-world conditions.

As networked systems become critical infrastructure, their design must reflect their new societal role. Today we build systems with hundreds of heuristics and do not understand their inherent and emergent behaviors. Most of us experience the resulting unreliability ("flakiness") every day when we use applications.

To scale performance proofs to real-world networked systems, I developed two ideas. First, prior work treats the network and its workloads as random objects. I treat them as non-random but, non-deterministic objects that can capture a large set of system behaviors in a succinct mathematical representation. Second, understanding heuristic behavior in a non-deterministic model requires combinatorial reasoning. I used automated verification to write proofs and to find counter-examples where widely deployed heuristics fail.

I have used this philosophy to obtain three key results in congestion control. These include a new algorithm that achieves high throughput while maintaining low network delays (Copa) [1], a proof tool to establish performance properties of various algorithms (CCAC) [2], and a theorem showing that current end-to-end approaches to controlling delays in networks cannot avoid starvation [3]. In my work on wireless communication, I built the largest published antenna array (3200 inexpensive antennas) ever used for a single communication link to overcome fundamental diffraction limits and beamform with high precision [4]. In my work on privacy-preserving computation, I built a cryptographically secure allegation escrow system [5] and contributed to Privid, a new way to analyze video datasets using black-box machine learning models while preventing the analyst from obtaining any private information from the video except aggregate results that preserve privacy [6].

The methods I developed to prove performance properties of network algorithms and protocols have sparked interest among researchers, with groups from CMU, UT Austin, Georgia Tech, MIT, Microsoft Research, and Waterloo all working on related projects. I am collaborating with several of them. It has also impacted industrial practice. For instance, the CCAC paper proposed a modification to BBR, a widely deployed CCA developed by Google. Facebook uses this modification for a *vast majority* of their user-facing traffic. In addition, for live video uploads, Facebook uses Copa [7].

Formalizing Congestion Control [1, 2, 3, 8]

Congestion control algorithms (CCAs) are distributed algorithms that decide how quickly a flow of data can be transmitted over a network. There has been a rapid pace of innovation in CCAs, driven by evolving network technologies, changes in the mix of applications, and the increasing importance of providing a good user experience. Poor performance can lead to negative reactions from users, such as giving applications low ratings or seeking out alternatives. It is not only average performance that matters but also the tail statistics. In response, the research community and industry have developed many innovative methods to improve congestion control. Large teams of engineers in the industry work on this problem.

Congestion control is challenging because real-world network paths exhibit a wide range of complex behaviors like wireless frame aggregation and channel quality variations, policers, end-host scheduling jitter, delayed acknowledgments, and more. It is impossible even for seasoned engineers to contemplate the composition of every "weird" thing that could happen along the path.

Over the past decade, numerous algorithms have been developed and deployed on the internet. However, they all experience unexpectedly poor performance on certain network paths, affecting users every day. As a community, we often do not understand why that happens. My goal is to develop methods that lead to a deeper understanding and use those methods to design better protocols.

To that end, I have made three contributions:

- CCAC [2]: I discovered a simple mathematical description that captures the effects of these complex real-world phenomena and built a tool, CCAC, that can verify performance properties of CCAs. The key idea is to express the model, the algorithm, and the property to be established in first-order logic and to use automated theorem provers to either prove the result or find a counter-example.

  Using CCAC we found previously unknown ways in which three widely deployed algorithms—AIMD, BBR and Copa—perform poorly. I was also able to use insights from the tool to modify BBR and prove that the modified version always converges to full utilization and bounded delay in our model. Facebook uses this version of BBR today for most of their user-facing traffic.

  The proofs produced by CCAC are the first to include the complex network behaviors discussed above. Prior work on CCAs has only proved properties on links whose rate is either constant or described by a simple stochastic process.

- Starvation [3]: Encouraged that I could use CCAC to formally prove the existence of an algorithm that could achieve high utilization with bounded delays, I turned my attention to multiple flows sharing a bottleneck link, expecting to formally prove that this modified BBR achieved fairness. Surprisingly, the tool found that not only was it unfair, but that it could be arbitrarily unfair—one flow could obtain all the bandwidth, starving the other flows. I then found that none of the other prior delay-bounding end-to-end protocols, including the original BBR and Copa, could avoid starvation

  Intriguingly, I found a property shared by all known delay-bounding end-to-end CCAs: when run on ideal constant bit-rate links, they all maintain a constant queue size, making small oscillations (if any) about an equilibrium. While this looks like an intuitive and even desirable property, I proved that any CCA with this property will definitely starve on some network path.

  Does starvation occur in the real world? If so, why hasn't it been observed before? It is hard to detect starvation in the wild because no entity has the visibility needed to correctly assign blame between the CCA and a genuine lack of network capacity. This may be why it has not been noticed before. Nevertheless, starvation is easy to reproduce even in simple settings. For example, BBR starves when the flows have different propagation delays and the network has some jitter of any kind. Both are extremely common on the internet.

- Copa [1]: I designed a new CCA, Copa [1], which incorporates two key ideas. First, it includes a filter that helps it distinguish queuing delay at the bottleneck from other sources of delay variations. Second, it detects whether it is sharing a link with a non-delay-sensitive CCA that is playing by a different, and incompatible, set of rules and adapts by transitioning to a more competitive mode. Facebook has been using Copa for live video uploads since 2019 because it provides better performance for its workload [7].

Beamforming with Thousands of Passive Antennas [4]

A radio's ability to direct its signal is fundamentally limited by its size. This limitation is acute on IoT and mobile devices, which are small and inexpensive, but even access points and base stations are typically constrained to a modest number of antennas. To address this limitation, I built a system, RFocus, with 3200 antennas. To the best of my knowledge, aside from radio telescopes and expensive military radar, this is by far the largest antenna array ever built. RFocus is practical and inexpensive because the antennas do not transmit or receive signals on their own; they are RFID-like backscatter devices arranged as a large ($\sim 6\, m^2$) wallpaper. Each has a single, two-state RF switch. RFocus helps devices in the vicinity of the wallpaper communicate with each other by configuring the antennas to reflect energy between them. In experiments, it improved the median signal strength by $9.5\times$.

Such devices, called Intelligent Reflective Surfaces (IRS) or metasurfaces, have been extensively studied using theory, simulations, and small-scale prototypes. Many applications have been theorized but not realized. RFocus is the first large prototype with over a thousand antennas, and in building it, I discovered a practical challenge. The contribution from individual antennas is approximately one million times smaller than the base channel. The paper's main contribution is a provably optimal method for measuring the channel and configuring antennas to increase signal strength using off-the-shelf radios. I am excited to continue this work to explore the new abilities and applications that thousands of inexpensive antennas can offer.

Privacy-Preserving Computation [5, 6]

An "allegation escrow" is a trusted party that collects allegations of crime and guarantees that they will only be revealed if a certain number of other allegations have been made against the same perpetrator. I have designed a secure version of this system that uses cryptography and distributes trust among multiple parties. One of the key technical contributions of my design is the ability to match allegations in constant time and allow each person making an allegation to set their own threshold for when the allegations will be revealed. To discourage false allegations, the identities of the people making the allegations are revealed along with the allegations, but are kept hidden until the threshold is met.

In a different project,[1] we observed that public video cameras can be a threat to privacy but also a valuable source of data. Examples include monitoring compliance with Covid masking guidelines, road safety, and pedestrian movement for marketing purposes. I developed a way to apply differential privacy to video data, which allows aggregate statistics to be revealed while protecting private information. Unlike typical applications of differential privacy, video data is unstructured, which posed a key challenge. My solution was to make realistic assumptions about the video that the system can use to enforce privacy for a wide range of queries.

Ongoing and Future Work

My future work is in three areas, all driven by the goal of building robust computer systems with proofs of performance.

Performance proofs for various networked systems:
I plan to expand on the ideas that have been successful in congestion control to a wide range of networked systems, including cluster provisioning, traffic engineering, video streaming, wireless networks, and processor scheduling. For example, in work with colleagues at UT Austin and Georgia Tech [9], we used techniques I developed for congestion control to analyze CPU schedulers. We uncovered subtle logical bugs in the Linux scheduler and proved new theorems about work stealing and shortest remaining time first scheduling.

Automatic synthesis of performant-by-construction algorithms:
Mathematical rigor will help us confidently develop complex controllers with high-dimensional inputs and outputs. For instance, video streaming can use forward error correction, retransmissions, and alternate compression methods (incremental compression, neural super-resolution-based compression, etc.). It could navigate tradeoffs between video quality, stall probability, CPU utilization, network utilization, and battery usage. Most deployed methods today consider only a subset of these factors since humans find it difficult, if not impossible, to reason through all of their interactions.

As a first step in this direction, I am working with colleagues at CMU [8] to investigate whether program synthesis can help construct provably performant congestion controllers, starting from human-supplied algorithm templates. To go beyond templates and automatically invent new concepts, I am working with colleagues at MIT[2] to use deep learning to learn congestion control policies. Most prior work on deep learning for systems uses system traces and simulators. However, learned algorithms fail when they encounter previ-

[1]Led by Frank Cangialosi in collaboration with Neil Agarwal, Junchen Jiang, Srinivas Narayana, Anand Sarwate and Ravi Netravali

[2]Pouyah Hamadanain, Arash Esafahany, Mohammad Alizadeh, and Hari Balakrishnan

ously unseen behaviors in the real world. We are using the network model I developed to train congestion controllers that are robust to adversarial network behaviors.

In the future, I intend to pursue research that combines deep learning and formal methods to design robust systems. Deep learning can invent new ideas and concepts, while formal methods can take those ideas to produce simple, provably robust heuristics.

Scalable and explainable formal reasoning methods:
Scalability: Methods I have used thus far can analyze individual heuristics. I will explore techniques that can scale to analyzing end-to-end systems with multiple interacting components.

Explainability: Unlike human-constructed proofs that provide intuition, automated proofs are often impossible for humans to inspect. Even if we trust the theorem-prover to be correct, it is hard to understand why the theorem is true. There is always a fear that the user may have encoded the theorem incorrectly, and so the proof is meaningless. I plan to explore methods that help users find problems in theorem encodings. At the heart of this problem is to come up with a mathematical definition of what a human-understandable explanation means.

This planned work on formalizing reasoning about computer system performance will give me insight on how empirical reasoning works. My hope is that this will lead to an understanding of empirical reasoning that is as satisfactory as our understanding of theorems and proofs.

References

[1] V. Arun and H. Balakrishnan, "Copa: Practical delay-based congestion control for the internet," in *USENIX NSDI*, 2018.

[2] V. Arun, M. T. Arashloo, A. Saeed, M. Alizadeh, and H. Balakrishnan, "Toward formally verifying congestion control behavior," in *ACM SIGCOMM*, 2021.

[3] V. Arun, M. Alizadeh, and H. Balakrishnan, "Starvation in end-to-end congestion control," in *ACM SIGCOMM*, 2022.

[4] V. Arun and H. Balakrishnan, "Rfocus: Beamforming using thousands of passive antennas," in *USENIX NSDI*, 2020.

[5] V. Arun, A. Kate, D. Garg, P. Druschel, and B. Bhattacharjee, "Finding safety in numbers with secure allegation escrows," *NDSS Symposium*, 2020.

[6] F. Cangialosi, N. Agarwal, V. Arun, S. Narayana, A. Sarwate, and R. Netravali, "Privid: Practical, privacy-preserving video analytics queries," in *USENIX NSDI*, 2022.

[7] N. Garg, "Evaluating copa congestion control for improved video performance `https://engineering.fb.com/2019/11/17/video-engineering/copa/`," in *Facebook Engineering Blog*, 2019.

[8] A. Agarwal, V. Arun, D. Ray, R. Martins, and S. Seshan, "Automating network heuristic design and analysis," in *ACM SIGCOMM HotNets*, 2022.

[9] S. Goel, B. Mikek, J. Aly, V. Arun, A. Saeed, and A. Akella, "Quantitative verification of scheduling heuristics," in *In Submission*, 2023.

[10] S. Narayana, A. Sivaraman, V. Nathan, P. Goyal, V. Arun, M. Alizadeh, V. Jeyakumar, and C. Kim, "Language-directed hardware design for network performance monitoring," in *ACM SIGCOMM*, 2017.

[11] A. Balasingam, K. Gopalakrishnan, R. Mittal, V. Arun, A. Saeed, M. Alizadeh, H. Balakrishnan, and H. Balakrishnan, "Throughput-fairness tradeoffs in mobility platforms," in *ACM MobiSys*, 2021.

[12] W. Sussman, E. Marx, V. Arun, A. Narayan, M. Alizadeh, H. Balakrishnan, A. Panda, and S. Shenker, "The case for an internet primitive for fault localization," in *ACM SIGCOMM HotNets*, 2022.

[13] Z. Meng, T. Wang, Y. Shen, B. Wang, M. Xu, R. Han, H. Liu, V. Arun, H. Hu, and X. Wei, "Enabling high quality real-time communications with adaptive frame-rate," in *USENIX NSDI*, 2023.