# Research Statement

**Ahmed Saeed**

Modern everyday activities are facilitated by real-time applications deployed on top of fast reacting ubiquitous computing platforms (e.g., smart assistants, online gaming, autonomous vehicles, and video processing). These user-facing applications are enabled by large-scale datacenter backends that process and store petabytes of data. Real-time applications enforce strict performance requirements on computing platforms both in the datacenter and at the edge. Meeting these requirements has become increasingly challenging because of a fundamental shift in the way hardware capacity scales. Rather than relying on Moore's law to deliver faster hardware every couple of years, modern systems have to leverage parallel processing and task-specific accelerators. My research at the intersection of **computer networks and systems** allows applications to meet their performance requirements, given this shifting hardware landscape.

Most of my work tackles the growing gap between the single-thread performance of a CPU and network speed. Specifically, over the past 10 years, single-thread performance doubled while network speed grew by 10×, putting tight constraints on software induced delays in processing network traffic. The network stack in today's operating systems is a remnant from a different time, when a single core could easily saturate the network. It simply cannot keep up with modern performance requirements. To improve the performance of the network stack, I designed efficient data structures for packet scheduling [NSDI '19] and efficient algorithms for admission and congestion control [SIGCOMM '20, OSDI '20, CoNEXT '19]. Further, my research is driven by the objective of improving real-world systems, requiring a deep understanding of deployed systems and their workloads. I developed my practical views of computer systems through years of hands-on experience and continuous collaboration with practitioners. *My work resulted in production-quality systems that are currently used as part of Google's infrastructure* [SIGCOMM '17].

My approach to research is to iterate between measurements, modeling, and implementation. Correspondingly, my first step in exploring a problem is to extensively and thoroughly measure the performance of existing schemes (e.g., collecting traces from real networks [SIGCOMM '17, SIGCOMM '20, TMA '19], analyzing publicly available datasets [ICNP '19], or benchmarking systems in the lab [OSDI '20, CoNEXT '19]). Measurements provide a clear understanding of the extent of the problem and the potential impact of solving it. Usually this leads to a suite of problems that I tackle sequentially, going back and forth between modeling the problem to gain a deep understanding of its root cause and implementing systems to validate my models. This rigorous process allows me to identify and tackle real problems with solutions that make a real impact.

The rest of this statement describes my work in three thrusts. The first body of work is on **scalable software network stacks (§1)**, solving problems in different components of operating systems and applications to allow a single server to handle tens of thousands of clients. The second body of work is on **Wide Area Network (WAN) congestion control (§2)**, focusing on network-assisted congestion control schemes, where end-to-end solutions fail. The third body of work is on **edge capacity and applications (§3)**, focusing on increasing and leveraging the network capacity at the edge.

## 1 Scalable Software Network Stacks

Over the last decade, the number of CPU cores on a server has increased significantly (from 8 to 128). Operators have in turn dramatically increased the number of clients simultaneously handled by a single server, putting significant strain on operating systems. In particular, managing and allocating system resources efficiently across a large number of clients is a key challenge. With current operating systems, the overhead of resource management tasks can easily overwhelm the system, leaving fewer resources for useful application work. The problem is particularly acute in the network stack, which must operate at very high frequencies to keep up with increasing network speeds. For example, a modern 100 Gbps server needs to schedule packets at nanosecond-timescales. My research tackles three essential operations in massively scalable servers: 1) packet scheduling, 2) scheduling applications' access to the network stack (managing egress traffic), and 3) scheduling clients' access to the server (managing ingress traffic).

**Packet Scheduling:** My work starts by observing that packet scheduling is the most CPU intensive operation whose cost grows with the number of clients. In my first project on this topic, Carousel [SIGCOMM '17], I proposed a different abstraction for rate limiting, the simplest scheduling policy. The idea behind Carousel is to enforce multiple rate limits using a single queue, rather than the conventional approach of having a queue and a token bucket per rate limit. This approach makes Carousel oblivious to the number of clients and their schedules. Specifically, applications tag packets with a rank, depending on the rate limit of their flows. The job of the scheduler is to sort packets according to their rank in a single queue. Carousel relies on a bucket-sort-based priority queue with $O(1)$ insertion and $O(1)$ deadline-based extraction of packets. The added benefit of an efficient rate limiter can diminish if access to the scheduler has to be synchronized between multiple CPU cores and applications are allowed to overwhelm the scheduler. Carousel employs a queue-per-core scheme in which queues are synchronized lazily. Further, Carousel applies backpressure to applications, regulating their access to the scheduler. Carousel reduces the overhead of the network stack by 20%. On 72-CPU machines, like machines that serve Youtube videos where Carousel is currently deployed, Carousel saves an average of 4.6 CPU cores per machine.

Rate limiting is critical for many applications but it is just one type of policy. Network operators use complex scheduling policies, like hierarchical fair queuing, to define the relative importance of different network users. My follow up research, Eiffel [NSDI '19], enables the efficient implementation of *any* scheduling policy. Creating efficient general-purpose schedulers is particularly challenging because it requires handling work-conserving schedules. Unlike a rate limiter, which can extract packets at predetermined times, a work-conserving scheduler must implement an `extract_min` operation to select packets based on arbitrary rank values (which are not based on time). I designed a bucket-sort-based priority queue that achieves this objective. The data structure represents bucket occupancy using a hierarchical bitmap, where zeros represent empty buckets and ones represent nonempty buckets. The implementation of the data structure leverages the Find First Set (`FFS`) instruction available in modern CPUs, to efficiently find the minimum nonempty bucket. Eiffel outperforms state of the art packet schedulers by 3-40$\times$ in terms of either number of cores utilized for network processing or number of clients given fixed processing capacity.

**Application admission control at the server:** As the number of clients increases, more applications compete for a server's egress bandwidth, enqueuing packets in the network stack awaiting transmission. When the number of enqueued packets exceeds the CPU or memory capacity of the stack, excess packets are dropped. Packet drops at the server waste CPU and increase latency. Congestion inside the server resembles network congestion in many ways. The key distinction between the two is the delay of the congestion signal. In networks, the congestion signal is delayed by at least the propagation delay between the congested network element and the sender, but it is instantaneous within a server. zD [CoNEXT '19] eliminates the overhead of congestion within servers, leveraging instantaneous signaling. zD is a new architecture for backpressure in which the network stack polls applications for packets, instead of applications enqueuing packets in the stack. Applications register with the stack if they have a packet to send. The stack considers processing that packet only when it has resources. Schedules are enforced by controlling the order and frequency at which the stack selects certain flows, moving the scheduler from a per-packet scheme to a per-flow scheme.

**Client admission control to the server:** As the number of clients increases, the server can be overwhelmed by requests from too many clients, creating another problem resembling network congestion. One possible solution is to allow individual clients to inform the server when they have demand, letting the server poll clients for requests when it has capacity. However, the overhead of messaging a large number of clients individually to obtain an accurate estimate of their demand becomes prohibitively large, especially for microsecond-scale requests. Breakwater [OSDI '20] instead regulates the access of clients to the server by *speculatively* admitting clients to send requests, essentially guessing which clients might have demand. In particular, the server maintains a list of active clients (which have recently had demand), and sends credits to a subset of them at random. Speculation about client demand avoids the CPU cost of accurate demand estimation, but it could lead to occasional bursts of incoming requests when the server underestimates the level of existing demand. Breakwater handles these bursts using a simple local queue management strategy that drops excess requests. The benefits of speculative admission in CPU overhead, throughput, and latency far outweigh the slight increase in request drop rate compared to prior overload control mechanisms.

## 2 WAN Congestion Control

Wide Area Network (WAN) congestion control is essential for the smooth operation of the Internet, determining how the network is shared when bandwidth demand exceeds capacity. Congestion control is typically implemented at the endpoints (i.e., senders and receivers) to avoid imposing any limitations or expectations on the network. Following this end-to-end approach in developing congestion control is beneficial because it keeps network elements simple and puts the pressure on traffic sources to regulate their transmission. However, relying only on endpoints ignores information available inside the network. Further, senders can detect congestion events only when they receive acknowledgements from receivers, delaying the detection of congestion by at least a full Round Trip Time (RTT) between the sender and the receiver. As hardware evolves, programmable network elements are becoming cheaper, creating opportunities to revisit the design of congestion control schemes, where network elements play a major role.

**Handling WAN bottlenecks in the Datacenter Network:** Datacenters are interconnected through high capacity WANs. Data exchanged between two machines in different datacenters has to go through the datacenter network of the source, then the WAN, and finally the datacenter network of the destination. This WAN traffic competes for bandwidth with traffic local to both datacenters. My research on this topic thoroughly characterizes issues arising from the competition between WAN and datacenter traffic based on measurements collected from Google's datacdenters [SIGCOMM '20]. Performance issues arise when datacenter traffic competes with WAN traffic near traffic sources, causing long delays and poor network utilization. The root cause of the problem is the difference in the timescale of operations between the two types of traffic. Specifically, the RTT of datacenter networks is typically just a few microseconds and the RTT of WANs can be tens of milliseconds. This large difference in timescales forces the fast-reacting datacenter algorithms to shoulder the full burden of handling congestion, lowering the throughput and increasing the delay of datacenter traffic. I developed Annulus, a congestion control scheme that allows WAN traffic to react at the same timescale as datacenter traffic. Annulus leverages smart switches that provide direct feedback messages to senders when they are congested, reducing the reaction time of WAN traffic sources. Annulus increases bottleneck utilization by 10% and lowers datacenter flow completion time by 1.3-3.5$\times$.

**Inter-Autonomous System Congestion Control:** The Internet is now flat. Content providers, like Netflix, are directly connected to Internet Service Providers (ISPs) through peering links. This peering happens through multiple physical connections. When one peering link is congested, content providers can choose to send traffic through an alternative link. Selecting the best alternative route requires knowing the bandwidth demand of the traffic flowing between the two networks, which is calculated based on the demand of clients and known only to content providers. It also requires knowing the best route to deliver the data to the clients, which is known only to the ISP. In current deployments, content providers select alternative paths without consulting ISPs. Unison [ICNP '19] is a novel approach for the interaction between content providers and ISPs, where ISPs provide content providers with preferred alternative paths when a peering link is congested. Unison leads to better user quality of experience and higher utilization of the ISP network.

## 3 Edge Capacity and Applications

Edge computing alleviates the pressure on the Internet core and reduces the latency of serving data to users, improving the reliability of the infrastructure and the performance of applications. My research in this area addresses several problems with the goal of increasing the network capacity at the edge. Further, I develop novel applications to leverage the new technologies available at the edge.

**Cost-efficient Wireless Broadband:** White spaces are the portions of the wireless spectrum allocated to TV broadcast but not used by TV broadcasters. Regulations allow for the unlicensed use of white spaces, offering valuable opportunities for high-speed wireless communication. However, accurate detection of white spaces, as mandated by government regulators, requires high-cost accurate spectrum sensing equipment that can detect the spectrum incumbents, even close to thermal noise. I developed Waldo [ICDCS '17], a distributed sensing scheme that can accurately detect white spaces using low-cost sensors. Waldo'a detection accuracy, using low-cost sensors, is comparable to the accuracy of high-cost spectrum analyzers.

**Low Latency WiFi:** Improving the latency of communication over WiFi links is critical for edge applications with tight latency requirements, like augment reality. Contention for medium access between multiple wireless devices can introduce tens of milliseconds of latency. I designed the Soft Token Passing Protocol (STPP) [ICNP '18] for coarse-grain medium access control. STPP reduces contention by allowing only the device with the token to use the wireless medium. STPP is tailored for the unreliable wireless medium, allowing the existence of multiple tokens to avoid starvation.

**Applications:** Most of my research is concerned with the digital infrastructure. However, I occasionally enjoy developing new applications. The first application I worked on uses commodity WiFi networks to detect and localize people that do not carry any devices [PerCom '12]. The idea behind the system is simple: when people move, they impact the behavior of wireless signals. The number and location of people in an area can be determined by measuring and tracking their impact on the wireless links in that area. The system can locate people through an entire floor down to three meters using no special hardware and only a few seconds of calibration. The second application is target surveillance using autonomous drones. Specifically, I solved the problem of visual monitoring of objects while taking their location, orientation, and width into account [IPSN '17]. With drones being the scarcest resource, I focused on the problem of minimizing the number of drones required to monitor a set of targets under such constraints and **derived a best-possible approximation algorithm to solve the problem.**

## 4 Future Work

At the core of my research is the objective of providing services at scale, subject only to delays due to the physical limitations of the hardware. Such performance expectations are now commonplace for both end users (e.g., an end user playing a game with a VR headset) and enterprise and large scale operators. Meeting these ever growing expectations requires systems and networks that can efficiently scale, taking into account every nanosecond of added latency. To that end, moving forward I intend to focus on the following areas:

**Massively Scalable Servers:** One of the ultimate objectives that my research has been building up towards is developing an end host that can serve large volumes of traffic (e.g., video) to millions of clients simultaneously. From a hardware perspective, this is becoming possible as the processing and networking capacity of servers grow. For instance, the Ethernet Technology Consortium has already launched the specification for 800 Gbps Ethernet. However, as my research has shown, the state of the art operating systems and network stacks can only handle tens of thousands of clients. The two-orders-of-magnitude jump in capacity will require addressing several lingering challenges. My preliminary measurements identify several bottlenecks in the design of the such massively scalable network stacks, including efficient management of state to avoid CPU and cache overhead. Further, application admission control will have to take the capacity of the CPU and memory as well as the processing capacity of accelerators in the network card. These considerations introduce new opportunities in protocol and data structure design as well as network function offload which I plan to pursue in my research.

**Edge Computing:** Bringing compute resources near end users is an integral part of planned cloud and telecom operators deployments. It is critical for applications that require massive compute resources at low latency (e.g., online gaming and autonomous vehicles). If working on datacenter congestion control has taught me anything, it is that connecting endpoints (e.g., clients and servers) through an ultra-low latency network does not guarantee that applications will always enjoy that low latency performance. Maintaining low latency network guarantees will require developing algorithms that can detect and resolve congestion fast in this new type of networks, while prioritizing critical traffic. Further, edge computing offers an exciting opportunity to customize the network stack based on the needs of the applications at the edge. The requirements of a health care application can be different from a video analytics application. Providing the best possible performance will require coordinating the behavior of servers, wired and wireless networks, and applications. I am well positioned to tackle these challenges as my research already touches on systems, networks, and applications.

**The Architecture of the Internet:** My earlier research has shown the promise and feasibility of having the network provide assistance with congestion control decisions. This is particularly useful when a network flow goes through significantly different types networks (e.g., a datacenter network and a WAN [SIGCOMM '20] and content provider networks and ISP networks [ICNP '19]). The growing heterogeneity of access networks (e.g., 4G, 5G, WiFi, wired-broadband, and satellite links) implies the rise of more such opportunities where network operations can be customized depending on the part of the network the data traverses. This will require innovations in expressing such heterogeneity as well as developing models and algorithms that can adapt to it.

**Leveraging Hardware Heterogeneity in Networked Systems:** Next generation servers will keep up with the slowing Moore's law by incorporating domain-specific accelerators (e.g., TPUs for machine learning and SmartNICs for networking). As we scale the loads on such servers and impose tighter latency requirements, software systems deployed on such servers will have to adaptively shift the load between different processors to make full use of all available processing capacity. Moreover, programmable switches can allow offloading some application logic from servers to switches. This will require innovation in the design of applications, systems, and scheduling algorithms to make use of all added processing capacity while minimizing the overhead of coordination.

Overall, I am broadly interested in systems and networking and looking forward to tackling other challenges that come my way in these fields.

## References

[SIGCOMM '20] **A. Saeed**, V. Gupta, P. Goyal, M. Sharif, R. Pan, M. Ammar, E. Zegura, K. Jang, M. Alizadeh, A. Kabbani, and A. Vahdat. Annulus: A Dual Congestion Control Loop for Datacenter and WAN Traffic Aggregates. In *Proc. of ACM SIGCOMM'20.*

[SIGCOMM '17] **A. Saeed**, N. Dukkipati, V. Valancius, V. The Lam, C. Contavalli, and A. Vahdat. Carousel: Scalable traffic shaping at end hosts. In *Proc. of ACM SIGCOMM'17.*

[NSDI '19] **A. Saeed**, Y. Zhao, N. Dukkipati, E. Zegura, M. Ammar, K. Harras, and A. Vahdat. Eiffel: efficient and flexible software packet scheduling. In *Proc. of USENIX NSDI'19.*

[OSDI '20] I. Cho, **A. Saeed**, J. Fried, S. J. Park, M. Alizadeh, and A. Belay. Overload Control for $\mu$s-scale RPCs with Breakwater. In *Proc. of USENIX OSDI'20.*

[CoNEXT '19] Y. Zhao, **A. Saeed**, E. Zegura, and M. Ammar. zD: a scalable zero-drop network stack at end hosts. In *Proc. of ACM CoNEXT'19.*

[IPSN '17] **A. Saeed**, A. Abdelkader, M. Khan, A. Neishaboori, K. A. Harras, and A. Mohamed. Argus: realistic target coverage by drones. In *Proc. of ACM/IEEE IPSN'17.*

[ICNP '19] Y. Zhao, **A. Saeed**, M. Ammar, and E. Zegura. Unison: Enabling Content Provider/ISP Collaboration using a vSwitch Abstraction. In *Proc. IEEE ICNP'19.*

[ICNP '18] **A. Saeed**, M. Ammar, E. Zegura, and K. Harras. If you can't beat them, augment them: Improving local wifi with only above-driver changes. In *Proc. of IEEE ICNP'18.*

[ICDCS '17] **A. Saeed**, K. A. Harras, E. Zegura, and M. Ammar. Local and low-cost white space detection. In *Proc. of IEEE ICDCS'17.*

[PerCom '12] A. E. Kosba, **A. Saeed** , and M. Youssef. RASID: A Robust WLAN Device-free Passive Motion Detection System. *IEEE PerCom'12.*

[TMA '19] K. Abdullah, N. Korany, A. Khalafallah, **A. Saeed**, and A. Gaber. Characterizing the effects of rapid LTE deployment: A data-driven analysis. In *Proc. of IEEE TMA'19.*