# Research Statement

**Ahmed Saeed**

*October 20, 2020*

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). At the backend of user-facing platforms, datacenter-scale applications 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 ever growing expectations became even more challenging because of a fundamental shift in the way hardware capacity grows. Rather than getting free speed ups, relying on Moore's law, 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 within the datacenter and at the edge, given the 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. To meet these requirements, 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 conversations 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 problems is to extensively and thoroughly measure the performance of existing schemes (e.g., collecting traces from real networks through collaborations [SIGCOMM '17, SIGCOMM '20, TMA '19], analyzing publicly available datasets [ICNP '19], or benchmarking systems in the lab [OSDI '20, CoNEXT '19]). Measurements give me a clear understanding of the extent of the problem and the potential impact of solving it. Usually, I end up defining 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.

To this end, my work follows three paths. 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 past years, the processing capacity of servers scaled out while their network capacity scaled up. Operators make use of the added capacity by increasing the number of clients simultaneously handled by a single server. As the number of clients increases, the operating system has to efficiently manage and balance the resources allocated to each client. Specifically, as the number of clients increases the resources needed to just manage them, not including the resources needed to actually serve them, can overwhelm the system, reducing the efficiency of the system. The overhead of managing clients is exacerbated by the need to perform resource allocation at very high frequency to keep up with increasing network speeds. For example, packet scheduling has to be performed 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 enforcing multiple scheduling policies using a single queue, rather than the conven-

tional approach of having a queue per policy. This approach makes Carousel oblivious to the number of clients and their schedules. Specifically, applications tag packets with a rank, depending on their scheduling policy. 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. The non-work conserving nature of rate limiting means Carousel never searches for the packet with the minimum transmission time, allowing for deadline-based $O(1)$ extraction of packets. Carousel is 6.4% more efficient than standard rate limiters implemented in the Linux kernel. 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 clients is critical for many applications (e.g., putting a cap on bandwidth of network users). However, complex scheduling policies like hierarchical fair queuing are useful to define the relative importance of different network users. My follow up research, Eiffel [NSDI '19], generalizes the functions of Carousel. Eiffel enables the efficient implementation of any scheduling policy. Creating efficient general purpose schedulers is particularly challenging because it requires handling non-work conserving schedules. Specifically, it requires a data structure with efficient `insert` and `extract_min` operations. I designed a bucket-sort-based priority queue that achieves those objectives. 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× 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, applications compete for network bandwidth at the server. Specifically, packets are queued in the server 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 in many aspects network congestion. My work on zD [CoNEXT '19] eliminates the overhead of drops by realising that the key distinction between network congestion and server congestion is the delay of congestion signaling. Network congestion signals are delayed by at least the propagation delay between the congested network element and the sender, leading to delayed reaction to congestion events. However, congestion signaling at the server can be done instantaneously, allowing servers to avoid drops completely. zD enables a zero-drop network stack by pausing traffic sources immediately when the server runs out of resources, and resuming them again when resources become available.

**Clients 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 at the server that resembles network congestion. The problem becomes even more challenging when the service time of requests is at a microsecond timescale, where the network delay is comparable to the service time of the request. With a large number of clients, the CPU cost of messaging clients individually to get an accurate estimate of their demand becomes prohibitively large. Breakwater [OSDI '20] regulates the access of clients to the server by speculatively admitting clients to send requests. To avoid overloading the server, the number of admitted clients changes depending on the state of the server, if the server is underutilized more clients are admitted, and when it is congested less clients are admitted. Speculation about client demand avoids the CPU cost of accurate demand estimation. Breakwater improves server goodput (the number of requests that meet their Service Level Objective) and reduces latency compared to the state of the art.

## 2 WAN Congestion Control

WAN congestion control is a method for regulating data entering the network. It is essential for the smooth operation of the Internet. Congestion control is typically implemented at the endpoints (i.e., senders and receivers) to avoid imposing any limitations or expectations on the network. The classic end-to-end principle in developing congestion control is beneficial for many reasons including keeping the design of network elements simple and putting the pressure on traffic sources to regulate their transmission. On the other hand, it is problematic as reaction to congestion has to happen at least after a full round trip time between the sender and receiver which can be tens of milliseconds long. As hardware evolves, programmable network elements become cheaper, creating opportunities to revisit the design of congestion control schemes. My research identifies scenarios where this end-to-end principle fails and proposes solutions that improve the reliability of the network with minimal deployment overhead.

**Handling WAN bottlenecks in the Datacenter Network:** As our reliance on cloud-based services grows, cloud service providers expand their infrastructure, including the WAN connecting their datacenters. WAN traffic traverses the datacenter network first before it reaches the WAN. As the bandwidth demand of WAN traffic grows, it is now possible to have the datacenter networks congested by WAN traffic. When WAN and datacenter traffic compete for bandwidth, both types of traffic suffer from performance degradation. My research on this topic thoroughly characterizes issues arising from the competition between WAN and datacenter traffic and proposes Annulus [SIGCOMM '20], a congestion control scheme that handles this type of congestion. My study of this problem is based on measurements conducted on Google's infrastructure and simulations used to validate our conclusions. The root cause of the problem is the difference in the timescale of operations between WAN traffic and datacenter traffic, which can be a 1000× difference. When congestion happens, an end-to-end WAN congestion control algorithm takes 1000 datacenter Round Trip Times (RTTs) to detect the congestion event, during that time the datacenter traffic shoulders the full burden of reacting to congestion. Annulus relies on two control loops. One control loop leverages existing end-to-end congestion control algorithms for bottlenecks where there is only one type of traffic (i.e., WAN or datacenter). The other loop handles bottlenecks shared between WAN and datacenter traffic near the traffic source, using direct feedback from the bottleneck, effectively reducing WAN feedback delay and improving its reaction time. I implemented Annulus on a testbed at Google and in simulation. Annulus increases bottleneck utilization by 10% and lowers datacenter flow completion time by 1.3-3.5×.

**Inter-Autonomous System Congestion Control:** The Internet is now flat. Content providers like Google and Amazon are directly connected to Internet Service Providers (ISPs) through peering links. This peering happens through multiple physical connections. Unison [ICNP '19] presents a novel approach for the interaction between content providers and ISPs to handle scenarios where some of the links connecting them become congested. Our analysis of congestion measurements between content providers and ISPs shows that when a peering link is congested, there are usually multiple other links with excess capacity. Selecting the best alternative route requires knowing bandwidth demand of 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. Unison allows content providers and ISPs to work together to select the best alternative route, as neither of them individually has enough information to select the best alternative route. 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 technology 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 the white spaces, offering a valuable new opportunity for high speed wireless communication. However, Accurate detection of white space, as mandated by the FCC, requires high cost accurate spectrum sensing equipment that can detect the presence of wireless signals, even close to thermal noise. To address this challenge, I developed Waldo [ICDCS '17], a distributed sensing scheme that can accurately detect white space. I implemented Waldo and used it to detect white spaces in the city of Atlanta, validating my results using high-cost accurate spectrum analyzers. Waldo improves the accuracy of low-cost sensors by 10×.

**Low Latency WiFi:** Improving the latency of communication over WiFi links is critical for edge applications with tight latency requirements like augment reality. I propose the Soft Token Passing Protocol (STPP) [ICNP '18] for coarse-grain medium access control. STPP is a token passing protocol tailored for the unreliable wireless medium, allowing multiple tokens to avoid starvation. STPP improves the performance of latency-sensitive applications as well as overall network performance.

**Applications:** Most of my research is concerned with the digital infrastructure. However, I enjoy occasionally working on developing new applications. I worked heavily on two particular applications. In the first project, I used 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 a three meters using no special hardware and only a few seconds of calibration. The second application is target surveillance using visual sensor networks. I used autonomous drones as the nodes in the visual sensor network. 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. Making that two-orders-of-magnitude jump will require addressing several lingering challenges. My preliminary measurements identify several bottlenecks in the design of the network stack 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 stack based on the needs of the applications at the edge. The performance 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 the 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.

**Heterogeneous Hardware:** Next generation servers will keep up with the slowing Moore's law by incorporating tasks 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 of its resources. This will require innovation in system design and scheduling algorithms.

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*.