

RESEARCH STATEMENT – Matthew Caesar

The Internet, composed of hundreds of millions of potentially-misbehaving hosts and thousands of competing ISPs, stands as one of the most complex and large-scale systems ever built by humankind. Concerns about the Internet's ability to meet ever-increasing demands on performance and functionality in the presence of this complexity has led to a call to redesign the Internet's architecture, for example in the context of the NewArch project, and NSF's GENI and FIND programs. A key challenge faced in designing a new Internet lies in *management* and *configuration*. These issues were overlooked when designing early data networks and the Internet has been paying a massive price ever since. ISPs hire armies of engineers to manually configure routers and debug problems, and in daily life we are surrounded by an ever-increasing array of complex embedded devices that require substantial configuration to interoperate. Forcing humans to configure and manage networks increases reaction time to faults, introduces the potential for misconfiguration, and substantially increases operating costs.

What is lacking today is a principled look at how to make systems manage themselves. We need a fresh approach to designing networks and protocols with self-management in mind. Toward this goal I propose to develop a class of *self-* protocols* that bootstrap, configure, and troubleshoot problems with only minimal manual intervention. In particular, these protocols aim to *self-configure* in the presence of arbitrary topologies and failure modes, *self-diagnose* routing problems, and *self-tune* operation based on diagnoses.

Prior work

As a first step in this direction, my thesis focused on the particular problem of building a self-managing network protocol. A self-managing network must perform two functions: assign identifiers to nodes and build routes between nodes. Today, network operators are heavily involved in these actions. To assign identifiers, operators must distribute IP address ranges to routers and end hosts, bind DNS names to these addresses, and assign access controls. To build routes, operators must configure policies in underlying routers, check for oscillations or other anomalous behavior arising from violations of policy correctness, and troubleshoot reachability problems. Hence, a self-managing routing protocol must automate both addressing and routing. My work deals with addressing through a new approach to routing that eliminates addressing entirely, and allows routers to forward packets directly on end-host names. To deal with routing, I developed a logically-centralized network-wide control plane that automates router configuration, performs route assignment on behalf of routers, and localizes faults and misconfigurations.

Future work

New network architectures:

New Internet architecture: The Internet is suffering a relentless inflation of routing update loads that shows no signs of slowing. Worse still, recent measurement studies show that this inflation is exceeding the capabilities of Moore's law, requiring nonlinear cost increases in router hardware to keep up. Hence today's architectural requirement of maintaining a complete table of all destination prefixes is quickly becoming impossible, especially in the presence of increased demands for deaggregation and the larger address spaces of IPv6. I have recently begun developing a more

scalable network architecture based on my thesis work that does not require a complete table. Instead of routing on flat *host* identifiers, we treat each *subnet prefix* as a numeric identifier. We then leverage a scheme based on my thesis work to self-organize and route between subnets by making progress in the prefix space towards the destination subnet. This approach retains several benefits of flat identifiers, yet has several deployability advantages.

Using randomization to build stable systems: We are developing techniques for improving stability of distributed systems by intelligently selecting which components to use. Our techniques are based on an observation called the *waiting time paradox*, which states that if interarrival times of failures have any variance, the expected waiting time to failure when selecting components randomly is biased towards longer times. We have found that incorporating randomization into component selection can drastically reduce sensitivity to faults. We are developing general techniques that leverage this insight to improve stability in several distributed systems, including path selection in wireless networks, and virtual machine migration in service clusters.

Convergence-free routing: Today’s Internet suffers from long outages arising from a slow convergence process that occurs after certain events. Slowing convergence was in fact done by design to improve stability: by rate-limiting updates, fewer route changes take place. Conventional wisdom is that this tradeoff is fundamental. I have recently begun exploring an alternate approach where instead of immediately propagating updates after a failure, we allow routing state to remain inconsistent, but provide a mechanism for data packets to autonomously discover a working path. Preliminary results show this approach reduces churn while maintaining extremely low loss rates.

Debugging/troubleshooting:

Wireless health monitoring: Wireless networks experience a vast array of failure modes which cause instability, starvation, and inefficiency. These problems are notoriously difficult to debug. To address these problems I aim to build a wireless health monitoring system. This tool leverages recent developments in phased array antenna technology to form a complete view of network operation from a single vantage point by passively observing wireless signals. We are currently developing a set of inference techniques that leverage observations from the system to localize problems. Such a tool can be used to diagnose faults, collect insights on how to reconfigure the network to eliminate hotspots, and detect malicious behavior and unauthorized connections.

Learning-based system management: Faults in distributed systems are currently a “fact of life”: software has bugs, operators make mistakes, and systems get attacked. However, troubleshooting faults and tuning system performance is currently a black art, relying heavily on heuristics. To formalize the management of systems, I aim to apply machine learning to model, predict, and diagnose system misbehavior. I am interested in applying statistical inference techniques to contain instability and localize faults in networks and component-based applications. My goal is to develop techniques to perform inference quickly, to speed diagnosis and reaction time.

Debugging across competitive domains: A key challenge in troubleshooting today’s large-scale distributed systems lies in coordinating the effort across multiple autonomous participants who may distrust each other or have privacy concerns. I am interested in leveraging privacy-preserving query protocols to troubleshoot faults across networks owned by different providers. In this fashion, participants can localize and route around faults without becoming aware of the specific root cause.