

My research goal is to enable distributed systems that are scalable and robust, maintaining fast and stable responsiveness at massive scales even when faults occur. Scale-out architectures have dominated the systems landscape over the last decade, with modern datacenters offering applications the promise of massive scalability and availability at very low costs. Delivering on this promise is a significant research challenge — datacenters consist of thousands of inexpensive fault-prone components, running commodity operating systems and protocols ill-fitted for high-performance applications. Further, datacenter applications have unconventional scaling requirements and bursty workloads that frequently push systems into delays and down-time.

My approach is to provide systems with low-level primitives for reliable communication that are fundamentally scalable and robust to faults and attacks. The protocols I design exploit the intrinsic properties of datacenter networks and are tuned to the usage patterns of clustered applications. An important aspect of my work is the use of *proactive* fault-handling techniques such as Forward Error Correction (FEC) and Gossip to achieve stable performance. Reactive protocols do too much too late, imposing extra overheads and delays that often send systems into spirals of degrading performance. In contrast, proactive protocols impose stable, predictable overheads and recover from faults almost instantly, preventing transient overloads and failures from translating into application unavailability.

Two systems that I have designed and built over the course of my PhD reflect these ideas. **Ricochet** (in NSDI 2007 [1]) is a low-latency messaging layer for clustered applications running *within* datacenters. **Maelstrom** (in NSDI 2008 [2]) is a transparent network appliance for reliable and rapid communication over high-speed optical networks *between* datacenters. Both protocols use fast and simple XOR operations in novel ways that allow datacenter applications to scale in new and vital dimensions. In particular, they create XORs at strategic points in the network (respectively, at multicast receivers and in an appliance) and from different data channels to obtain excellent recovery and latency properties. Together, these protocols allow for the development of highly available applications that coordinate within and across datacenters while maintaining extremely scalable and robust responsiveness.

1 Reliable Low-Latency Messaging for Clustered Applications

The Problem: Reliable multicast (RM) is at the heart of commercial datacenter software, used by publish-subscribe layers, clustered application servers and distributed caching infrastructures. The dominant failure mode for datacenter multicast is packet loss at overloaded end-hosts, when kernel buffers drop packets in short bursts. Most real-world multicast deployments involve hundreds of fine-grained groups, each of which has low and sporadic data rates even though the aggregate traffic in the system is high; an example is a financial pub-sub system that maps stocks to groups. Existing RM protocols are fundamentally unscalable in the number of groups in the system; some require expensive per-group structures, while others only work in groups with high, stable data rates.

The Solution: Ricochet uses a novel technique called *Lateral Error Correction* (LEC) to rapidly recover multicast packets dropped at end-host kernels. In LEC, receivers exchange XORs of incoming data in common groups — hence, recovery latency depends on the aggregate data rate in the entire system rather than the data rate in any one group. Ricochet is insensitive to patterns of usage and scales in the number of groups in the system; it provides the same performance in one dense high-rate group as in hundreds of fine-grained low-rate groups.

2 Transparent Error Correction for Lambda Networks

The Problem: Organizations are increasingly deploying global networks of datacenters interconnected by high-speed optical links. Running loss-free networks over such links is difficult and expensive; packet loss on the end-to-end path can occur for a number of reasons, including malfunctioning or misconfigured equipment and dirty or degraded fiber. Commodity TCP/IP performs very poorly on long-haul links — first, it backs off too sharply to low non-congestion loss, and second, it takes a minimum latency of an RTT to recover lost packets, inducing delays that may be untenable for high-performance applications. Many high-speed TCP/IP variants have been proposed that solve the first of these problems but not the second; in any case, deploying such variants is impossible in datacenters where standardization is key to mitigating costs.

The Solution: Maelstrom is a network appliance that resides between the datacenter end-hosts and the optical link and transparently inserts FEC packets into outgoing streams. When loss occurs on the link, a corresponding appliance in the receiving datacenter recreates missing packets from the FEC information. Maelstrom aggregates multiple flows for high-speed encoding and is completely transparent, requiring no modification to end-hosts or to the long-haul network. To combat bursty loss patterns, Maelstrom uses a new FEC scheme called layered interleaving that degrades gracefully against increasing burst sizes. A single-machine version of Maelstrom running in-kernel can handle up to half a Gbps of traffic; with simple load-balancing over multiple boxes, Maelstrom can encode over tens of Gbps of traffic.

3 Other Work

Ricochet and Maelstrom are real systems, with freely available implementations shown to scale massively on datacenter testbeds. In addition to being stand-alone systems, they have served as umbrella efforts within which other researchers in my group have built new abstractions, applications and extensions. The techniques used in Ricochet have led to ideas for low-latency message ordering (Plato [3], in SRDS 2006), been extended to mobile ad-hoc networks (Mistral [4], in Mobihoc 2006) and layered underneath higher-level service abstractions (Tempest, in submission). Similarly, Maelstrom has inspired new ideas for remote mirroring (SMFS, in submission) and power-saving filesystems (KyotoFS [5], in HotOS 2007), and is currently being extended into a scalable IPTV platform. My role in these other systems has included ideation and design; more importantly, my own system-building efforts have provided a coherent vision and concrete direction for other projects. In addition to driving collaborative systems research I have also enjoyed working on the practical aspects of theoretically motivated work, building a system at Microsoft Research for the tree-based modeling of wide-area latencies (Sequoia [6], in PODC 2007) and helping theoreticians at Cornell apply scheduling bounds to real-world problems in mobile ad-hoc settings (blinded, in submission).

4 Directions for future research

My goal for the future is to architect the next-generation datacenter — one that is energy-efficient, exploits new hardware designs such as multi-core chips, and acts in close orchestration with global ‘lambda’ networks of peer datacenters.

Lambda Networks: Lambda Networks provide rich and interesting opportunities — to provide resilience against natural disasters and terrorist attacks by mirroring data to far-away datacenters, to save power and money by locating primary copies of data in these remote datacenters, and ultimately, to create truly global applications that can run seamlessly across geographies. The combination of powerful computing clusters and massive high-speed links opens the door to ubiquitous data access, anytime and anywhere. A potential killer app for lambda networks is IPTV; I am currently designing a clean-slate IPTV architecture where datacenters located in different cities are

interconnected via lambda links and fiber-to-home provides optical last hops. Such a design raises many research questions — for example, the usage of end-to-end FEC to increase last hop range/receivers, or the implementation of scalable channel switching and video-on-demand.

Green Systems: Designs that treat power efficiency as a first-class goal are the need of the hour, both in light of organizational facility budgets and long-term environmental considerations. One approach is to place data and computation intelligently within and across datacenters. For example, I am collaborating with filesystem researchers at Cornell to create KyotoFS, a filesystem for highly cache-able workloads that organizes data in a distributed multi-disk log within a datacenter; since all writes go to the head of the log and reads can be serviced by in-memory caches, the disks in the middle and tail of the log can be switched off to save power. A parallel direction of research involves reducing the power consumption of multi-core systems; the availability of power-saving modes in newer chips allows systems and protocols higher up in the stack to explore trade-offs between performance and energy.

Datacenter Architecture: The slowdown in CPU frequency scaling and the emergence of multi-core machines creates new problems and opportunities for datacenters. Current datacenter systems are designed for settings where the unit of scale is an individual processor with dedicated IO bus-lines; in the future, such systems will have walk a tight-rope between intra-machine resource contention and inter-machine communication inefficiencies. Datacenters will evolve into clusters of clusters, requiring parallel scale-up techniques within many-core machines and distributed scale-out techniques between them. Meanwhile, concerns such as power, ISP non-neutrality and privacy are redefining the role of datacenters. A design where the datacenter hosts a control plane but all sensitive data is stored in the client cloud tackles such concerns; but implementing real applications like social networks or e-mail in this fashion is a non-trivial challenge.

To summarize, scale-out architectures have reached an inflexion point in their evolution where they face multiple challenges — real-time coordination across distant datacenters as well as power, privacy and parallelism within datacenters. My agenda for the future includes the design and implementation of systems that explicitly target these goals and effectively use high-bandwidth networks to achieve them.

References

- [1] Mahesh Balakrishnan, Ken Birman, Amar Phanishayee, and Stefan Pleisch. Ricochet: Lateral error correction for time-critical multicast. In *NSDI 2007: Fourth Usenix Symposium on Networked Systems Design and Implementation*, 2007.
- [2] Mahesh Balakrishnan, Tudor Marian, Ken Birman, Hakim Weatherspoon, and Einar Vollset. Maelstrom: Transparent error correction for lambda networks. In *NSDI 2008: Fifth Usenix Symposium on Networked Systems Design and Implementation (To Appear)*, 2008.
- [3] Mahesh Balakrishnan, Ken Birman, and Amar Phanishayee. Plato: Predictive latency-aware total ordering. In *SRDS 2006: IEEE International Symposium on Reliable Distributed Systems*, 2006.
- [4] Stefan Pleisch, Mahesh Balakrishnan, Ken Birman, and Robbert van Renesse. Mistral: Efficient flooding for mobile ad-hoc networks. In *MobiHoc 2006: 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2006.
- [5] Lakshmi Ganesh, Hakim Weatherspoon, Mahesh Balakrishnan, and Ken Birman. Optimizing power consumption in large scale storage systems. In *HotOS XI: 11th Workshop on Hot Topics in Operating Systems*, 2007.
- [6] Ittai Abraham, Mahesh Balakrishnan, Fabian Kuhn, Dahlia Malkhi, Kunal Talwar, and Venugopalan Ramasubramanian. Reconstructing approximate tree metrics. In *PODC 2007: 26th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, 2007.