Research Statement

Tudor Marian

My research interests span the areas of networking, distributed systems, and operating systems. Within these areas, my work revolves around the performance, fault-tolerance, and reliability issues that arise within the modern datacenter. My work also has a strong system building component based on solid engineering.

Over the past decade, the modern datacenter has taken the center-stage as the dominant computing platform that powers most of today's consumer online services, financial, military, and scientific application domains. Further, datacenters are becoming a commodity themselves, and are increasingly networked with each other through high-speed optical networks for load balancing (e.g., to direct requests to closest datacenter or provide localized services) and fault tolerance (e.g., to mirror data for protection against disasters). As a result, virtually every operation within and between datacenters relies on the networking substrate, making the network a first-class citizen. However, datacenters are built from inexpensive, fault-prone components running commodity operating systems and network protocols that are ill-suited for fault-tolerant, reliable, or high-performance networked applications. My research focuses upon the examination and improvement of the commodity datacenter's communication substrate.

I approach research problems through the principled and innovative use of analysis, measurement, abstractions, and algorithms. I strive to understand and tackle problems in a rigorous fashion, and validate the comprehension through measurements, design, artifact implementation, and system evaluation [1]. To date, I have applied this technique to a broad spectrum of problems: providing operating system support for scalable user-space packet processing at line (10Gbps) speed [2]; performing precise temporal characterization of 10Gbps optical wide-area networks [3] and studying the commodity endhosts connected to them [4]; improving the quality of wide-area network channels through performance-enhancement protocols [5, 6, 7] and disaster-tolerant efficient remote data mirroring [7]; and delivering a scalable, reliable, distributed platform for service oriented architectures [8, 9, 10].

1 Previous Work

Packet Processing Abstractions: Despite the fact that the network substrate is an essential part of the datacenter, existing conventional network protocols and commodity operating systems remain ill-suited for reliable, high-performance applications. Therefore, new network protocols are constantly being developed in order to provide large-scale coordinated services. However, operating system support for building network protocols in general, and *packet-processing* network protocols in particular, continues to lag. Developers are typically forced to trade off the programmability and extensibility of applications built to run on commodity computing platforms (e.g., general purpose servers with several network interfaces) for the performance (in terms of data and packet rate) of dedicated, hardware solutions, or low-level in-kernel implementations.

NetSlice [2] extends the operating system with a new abstraction that developers can use to build high-performance packet-processing protocols in user-space without incurring the performance penalty that conventional abstractions engender. Unlike the conventional operating systems abstractions, like the raw socket, NetSlice is designed to tightly couple the hardware and software packet processing resources, and provides the application with control over these resources. To reduce overall contention, NetSlice performs coarse-grained spatial partitioning (i.e., exclusive, non-overlapping, assignment) of processor cores, memory, and network interface controller (NIC) resources, instead of time-sharing them. Moreover, NetSlice provides a streamlined communication channel between NICs and user-space. NetSlice exports a highly efficient, yet also backward compatible, application programming interface (API) that allows batched (multi-) send / receive operations to amortize the cost of protection domain crossings. For example, complex user-space packet processors, like a protocol accelerator and an IPsec gateway, built with NetSlice and running on commodity components, can scale linearly with the number of cores and operate at nominal 10Gbps network line speeds.

Characterizing 10Gbps Optical Networks: High-bandwidth, semi-private optical lambda networks carry growing volumes of data on behalf of large data centers, both in cloud computing environments and for scientific, financial, defense, and other enterprises. Although lambda networks have sufficient bandwidth, are dedicated for specific use, and operate with virtually no congestion, end-hosts and applications find it increasingly harder to derive the full performance they might expect. The end-to-end characteristics of commodity servers communicating via high-bandwidth, uncongested, and long distance 10Gbps optical networks have not been well understood.

We performed a careful examination of the end-to-end characteristics of an uncongested lambda network running at high speeds over long distances, identifying scenarios associated with loss, latency variations, and degraded throughput at attached end-hosts [4]. Using identical commodity source and destination server platforms, hence expecting the destination to receive more or less what was sent, we observed otherwise: degraded performance was common and easily provoked. The receiver exhibited an impedance mismatch (at the interface between the NIC hardware and the system software) and packets were lost even when the sender employs relatively low data rates. Further, packet loss increased with the increase in path length.

Given the modest data rates the commodity servers drop packets at, we hypothesized that even on uncongested lambda networks, the forwarding routers artificially distort the flow of packets. To comprehensively characterize this phenomenon, we built a Software Defined Network Adapter (SDNA) [3]. The SDNA transmits packets by using a pulse pattern generator to modulate a laser's output with software-generated physical layer symbol-streams, and receives packets by sampling the signal with an optical oscilloscope. By relying on the precisely calibrated time-base of the test instruments, SDNA achieves six orders of magnitude improvement in packet timing precision over endpoint-software measurement and two to three orders of magnitude relative to prior hardware-assisted solutions. The SDNA revealed that an input flow with packets homogeneously distributed in time becomes increasingly perturbed as it traverses an uncongested network, to such an extent that within a few hops the egress flow becomes a series of minimally-spaced packet chains. This phenomenon occurs irrespective of the input flow data rate, and has severe consequences: traffic issued in-profile overwhelms receiving end-hosts with bursts of high-rate data that overrun buffers and cause loss.

Middleboxes Connecting the Network of Datacenters: Connecting datacenters over vast geographical distances is challenging, especially when relying on traditional communication protocols [4]. A common technology used to address this problem are perimeter middleboxes that perform packet processing to implement some form of performance enhancement network protocol. Maelstrom [5, 6] is such a network appliance located between the datacenter endhosts and the wide-area optical link, that transparently inserts forward error correction (FEC) packets into outgoing streams. When loss occurs on the link, a corresponding appliance situated at the receiving datacenter regenerates the missing packets from the redundant data. Maelstrom handles bursty loss patterns through a novel FEC scheme called layered interleaving, which gracefully degrades the FEC recovery as the loss burst size increases.

The Smoke and Mirrors File System (SMFS) [7] also relies on a middlebox and was inspired by the prior work on Maelstrom. SMFS is a remote mirroring file system built with disaster tolerance in mind that takes advantage of an edge appliance to provide a middle ground between synchronous (or semi-synchronous) and asynchronous mirroring modes. The SMFS middlebox *exposes* the status of the outbound data and redundancy traveling over the network. Relying on this knowledge, the sender can resume activity as soon as a desired level of in-flight redundancy has been achieved for any given packet, or point in a data stream—for example, when the probability of losing data renders the transmission link as reliable as a local disk.

Automatically Scaling out Service Oriented Architectures: Both the Scalable Services Architecture (SSA) [9] and the Tempest [8] projects were motivated by the observation that large computing systems, structured as ensembles of coordinated services, require to be robust when facing challenging operational conditions such as load surges, transient disruptions and failures, and are difficult to build. Furthermore, existing platforms rely on transactional database solutions for load-balancing and recovery, while developers of non-transactional services are left with implementing their own replication, failure detection, recovery, and load balancing mechanisms. For many services, the transactional model is a poor-fit. SSA and Tempest demonstrate that a simple, yet remarkably inexpensive infrastructure based on primitive mechanisms—epidemic protocols and a variant of chain replication—can support efficient clustered execution of a significant class of non-transactional services. The Tempest runtime relies on these primitives to provide storage and replication of service-level soft state, by providing *collection* data structure abstractions that are transparently replicated for availability and fault tolerance with no effort from the developer.

2 Future Work

In the future, I wish to provide datacenter operators and developers with systems and services that allow them to control and leverage the next-generation network of datacenters. Ideally, such systems would harness the modern commodity hardware designs, like multi-core processors and high speed network adapters, and leverage lambda network pathways that connect peer datacenters. Furthermore, due to the enormous scale at which networks of datacenters operate, power awareness and fault-tolerance should be first-class design principles. There are several key challenges that I plan to address in the near future: **(i)** How do we strike a balance between processing speed and energy utilization for the applications resident in the datacenter? **(ii)** How do we utilize the datacenter networking substrate effectively; the datacenter network layer is currently power-hungry yet largely underutilized? **(iii)** How does one continuously monitor the scores of applications running on numerous machines in the datacenter environment, analyze this data, diagnose problems, and react accordingly—all in an automatic fashion?

Energy consumption in modern datacenters has recently received an enormous amount of attention because of the growing importance of leaner operational budgets and long-term environmental impact. I have begun to collaborate on developing a storage system for the datacenter designed to save power by significantly reducing the number of disks that are kept spinning. The idea is to overlay a log abstraction over a fault-tolerant mirrored multi-disk array. Consider such a storage system on top of ten disks, of which five are used as primaries and five as mirrors (typical of a RAID–1 mirroring scheme without parity or striping). A log-structured storage system writes only to the log head, which means it continuously writes to the same two disks for long periods of time (for as long as it takes to fill these disks). Therefore, the storage system has the opportunity to power down the four remaining mirror disks. Read requests are served from the primary disks that are still powered up, trading off read throughput for power savings. Further, periodically cleaning up the tail of the log, which requires spinning up the mirror disks, is not on the critical path, hence the storage system may incur the large latency penalty due to powering up disks on demand. Importantly, the system ensures stable read and write throughput even during log cleanup. In particular, stable write throughput is achieved since the head and the body of the log are placed on different disks, while stable read throughput is ensured by serving reads from one mirror while the other is being cleaned.

Since power consumption in the datacenter is of paramount importance, I plan to explore power saving opportunities across the entire spectrum of components, and at various layers. For example, current network protocols and equipment were not designed with power awareness in mind, hence they are not always efficient. Particularly, wired networks, like 10GbE, transmit constantly on the wire at the maximum symbol rate, even though most physical layer information is *idle symbols* with the occasional data packet sandwiched in between (unless transmitting at a data rate close to maximum). This means that except for time intervals during which packets are sent at close to maximum symbol rate, the network elements are expending energy to encode idle symbols into frames. Conversely, the network elements also expend energy to decode frames into symbols that are predominantly idle. The energy proportionality of network equipment is far from ideal—when the amount of wattage expended is proportional to the load.[1] Recently, I have been involved in the development of a software defined network adapter [3] (SDNA) apparatus which, amongst others, allows us to rapidly prototype and implement in software new physical medium dependent (PHY) layers. I plan to leverage the SDNA to investigate and explore the feasibility of alternative, more power efficient PHY layers.

Monitoring the operations of tens of thousands of commodity servers is a daunting task. Further, analyzing performance logs and fault traces from such a large number of machines is challenging to say the least. I have begun to design and prototype Fmeter—a monitoring framework that yields low-level indexable descriptions, or signatures, of systems at runtime. Fmeter is sufficiently lightweight to continuously operate on deployed production systems with near-zero overhead. Unlike standard lightweight monitoring techniques that perform statistical profiling based on sampling (e.g. `oprofile`), Fmeter generates signatures analogous to the *term frequency–inverse document frequency* (*tf-idf*) weights used in information retrieval and text mining. In particular, Fmeter computes the tf-idf-like weights by treating the system's kernel *function calls* during an *execution*, as they would be *terms*, or words, within a *document.* This means that the signatures—essentially vectors of normalized kernel function calls frequencies—are amenable to be manipulated by conventional formal methods like clustering, classification through machine learning, and similarity based search against a database of previously labeled signatures. Of particular interest is automatically ap-

[1] For example, a Juniper EX8200 Ethernet switch consumes nearly 10kW, roughly as much as an entire rack of servers.

plying such techniques over the enormous volumes of report-data generated by a large number of machines, in order to pinpoint normal and anomalous behavior. Moreover, one can envision such classification tasks providing feedback into an automated liveness / reconfiguration service.

To summarize, my research will be geared towards building the next generation systems and services for the network of datacenters. I plan to focus my future research on improving the scalability, reliability, and the power-awareness of the clusters consisting of thousands of fault-prone, inexpensive machines which currently run commodity operating systems and protocols that are currently ill-suited for such an environment. Further, I wish to enable datacenters to coordinate among each other across large distances, while effectively leveraging modern computing hardware and high-bandwidth optical networks.

References

[1] **Tudor Marian**. *Operating Systems Abstractions for Software Packet Processing in Datacenters*. PhD Dissertation, Cornell University, Department of Computer Science, August 2010.

[2] **Tudor Marian**, Ki Suh Lee, and Hakim Weatherspoon. NetSlices: Scalable Multi-Core Packet Processing in User-Space. In Submission.

[3] Daniel A. Freedman, **Tudor Marian**, Jennifer H. Lee, Ken Birman, Hakim Weatherspoon, and Chris Xu. Exact Temporal Characterization of 10 Gbps Optical Wide-Area Network. In *Proceedings of the 10th Internet Measurement Conference (IMC' 10)*, November 2010.

[4] **Tudor Marian**, Daniel Freedman, Ken Birman, and Hakim Weatherspoon. Empirical Characterization of Uncongested Lambda Networks and 10GbE Commodity Endpoints. In *Proceedings of the 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN-PDS '10)*, June 2010.

[5] Mahesh Balakrishnan, **Tudor Marian**, Ken Birman, Hakim Weatherspoon, and Einar Wollset. Maelstrom: Transparent Error Correction for Lambda Networks. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI '08)*, April 2008.

[6] Mahesh Balakrishnan, **Tudor Marian**, Ken Birman, Hakim Weatherspoon, and Lakshmi Ganesh. Maelstrom: Transparent Error Correction for Communication between Data Centers. *To appear in IEEE/ACM Transactions on Networking (ToN)*, 2010.

[7] Hakim Weatherspoon, Lakshmi Ganesh, **Tudor Marian**, Mahesh Balakrishnan, and Ken Birman. Smoke and Mirrors: Shadowing Files at a Geographically Remote Location Without Loss of Performance. In *Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST '09)*, 2009.

[8] **Tudor Marian**, Mahesh Balakrishnan, Ken Birman, and Robbert van Renesse. Tempest: Soft State Replication in the Service Tier. In *Proceedings of the 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN-DCCS '08)*, June 2008.

[9] **Tudor Marian**, Ken Birman, and Robbert van Renesse. A Scalable Services Architecture. In *Proceedings of the 25th IEEE Symposium on Reliable Distributed Systems (SRDS 2006)*. IEEE Computer Society, 2006.

[10] Ken Birman, Mahesh Balakrishnan, Danny Dolev, **Tudor Marian**, Krzysztof Ostrowski, and Amar Phanishayee. Scalable Multicast Platforms for a New Generation of Robust Distributed Applications. In *Second International Conference on Communication System Software and Middleware (COMSWARE 2007)*. IEEE Computer Society, 2006.