conference reports

THANKS TO OUR SUMMARIZERS

NSDI '08: 5th USENIX Symposium on Networked Systems Design & Implementation

San Francisco, CA
April 16–18, 2008

KEYNOTE: XEN AND THE ART OF VIRTUALIZATION REVISITED

Ian Pratt, Senior Lecturer, University of Cambridge Computer Laboratory, and Fellow, King's College Cambridge

Summarized by Geoffrey Werner-Allen (werner@eecs.harvard.edu)

Mr. Pratt presented a talk in three parts. He began by revisiting the Xen story, presenting lessons concerning doing relevant research in academia. Next he explored why virtualization is such a hot topic in research today. Finally, he explored the changes in Xen since the 2004 SOSP paper and emerging trends in hardware-software co-design.

Although Xen emerged naturally from the cloud-computing ethos and the needs of usage-based accounting, Mr. Pratt pointed out that there was a significant pause in its early days, primarily because the funding agencies had nothing to compare their project to. During this period, Xen was release via the GPL to "friends and family" and their team began to notice the differences in doing development not for an academic paper, that is, when their creations had to run for more than the 30 minutes required to produce the graphs for the latest paper. They kept working on the production-level aspects of their software.

The mission of Xen is to provide the industry-standard open-source hypervisor. Xen developers are interested in driving CPU development and showcasing new CPU features, as well as providing the type of security necessary for enterprise acceptance. As Xen has continued to develop it has found use in some interesting and perhaps unforeseen areas, such as cell phones. Mr. Pratt described plans that one cell-phone vendor has to run three separate hypervisors on its phone to isolate critical phone functionality, vendor-supplied software, and user-installed software from each other.

Why is virtualization hot at this particular moment? One reason is that it is driven by the "scale out" occurring at the enterprise level. Running each enterprise application on a single server leads to server sprawl, with CPU utilization of 5%–15% typical. Another reason involves the things that typical operating systems have failed to do well, including full configuration isolation, temporal isolation for performance, spatial isolation for security, and true backward compatibility. Virtualization has the potential to solve many of these problems. Moreover, the maintenance of a narrow interface to the hypervisor and the ability to hide machine-specific details behind

the hypervisor allows easier configuration. Other benefits of virtualization include reduced downtime during maintenance owing to the ability to migrate slices to other machines, the ability to rebalance load as workloads change, and hardware fault tolerance through checkpointing and replay.

Mr. Pratt discussed the issues concerning hypervisor security. Although the existence of a hypervisor does add to the attack surface, its small size should make it easier to defend. He discussed network control software that their group has been writing in OCaml, a language that allows certain guarantees to be made about the program's execution. In addition, there is additional complexity involved in device emulation, so to the degree that this is required this increases the complexity required in the hypervisor. He also discussed the possibility of performing a "measured launch" of the hypervisor to provide guarantees between boots. In addition, moving some of the OS administrative functionality outside the OS should help operating systems become more robust and harder to disable. Other tricks that can be used to improve hypervisor security include the use of immutable memory and taint tracking techniques (detailed elsewhere).

The next frontier in virtualization research should be making virtualized systems easier to administer. In particular, breaking the OS/HW bond should simplify the application certification process. Currently, many software vendors refuse to certify their applications on many hardware configurations or in the presence of other applications. Instead of certifying an application on top of many different hardware and operating system combinations, software makers can certify on a single operating system, which is then certified on top of Xen, meaning that the application can run on a variety of different hardware configurations. We are already starting to see application-specific operating systems being developed in the presence of virtualization techniques such as Xen. Virtual hardware also simplifies creating and modifying operating systems, as well as allowing hardware vendors to "light up" new features much faster.

In the final part of his talk Mr. Pratt discussed the process of paravirtualization through several examples, including MMU and network device virtualization. Interestingly, with regards to MMU virtualization Xen proved that some of the support that hardware vendors have added into their systems, in particular allowing "nested page tables," have actually not performed as well as the original direct and shadow page tables supported by Xen.

Network devices have proved extremely difficult to virtualize, but new NICs are emerging with features that make the process simpler and more effective. Xen developers have divided NICs into four levels, 0 through 3, each with increasing features allowing easier virtualization with better performance. Level-0 NICs are standard NICs, requiring significant hypervisor intervention to virtualize. Level-1 NICs have multiple receive queues, allowing these queues to be assigned to VMs making heavy use of the network. Level-2 NICs have enabled direct guest access, allowing the NIC to do traffic shaping, firewalling, filtering, and other processes. Level-3 NICs actually present to the system as multiple PCI devices, simplifying device management at the hypervisor level.

In conclusion, Mr. Pratt recommended the use of open source software to gain early and continuing impact while doing university research. In the future he sees hypervisors becoming ubiquitous and spawning a new golden age of operating system research.

Professor Birman from Cornell was curious about the tension between adding features, making the hypervisor more like larger buggier operating systems, and consolidation, possibly impacting performance but improving security. Mr. Pratt pointed out that many of the new features he described are actually being implemented outside of the core hypervisor, meaning that they can be isolated from it for the purposes of security. Greg Minshall from the University of Washington asked about Xen's impact on the previous optimization that had been done at the hardware/software boundary. Mr. Pratt responded that as machines get faster this sort of optimization is less important and that Xen does as little of this as possible. Professor Sirer from Cornell asked why operating systems are unable to provide "virtualization" at the POSIX level. Mr. Pratt replied that the interface is simply too broad and high level. Tomas Isdal asked whether Xen developers had a plan to scale to multiple cores; Mr. Pratt replied that they did.

TRUST

Summarized by Eric Hielscher (hielscher@cs.nyu.edu)

- ***One Hop Reputations for Peer to Peer File Sharing Workloads***
  Michael Piatek, Tomas Isdal, Arvind Krishnamurthy, and Thomas Anderson, University of Washington

Michael began by explaining that peer-to-peer scalability depends on user contributions but that users are often reluctant to contribute. Current peer-to-peer systems explicitly include contribution incentives such as tit-for-tat servicing. Next he presented results of a measurement study that show that increasing one's contribution to BitTorrent swarms has very little effect on one's download rates. Further, there is no fundamental lack of capacity in the swarms nor a lack of interest. Rather, results from another study show that the vast majority of bandwidth capacity in BitTorrent swarms is held by the top 10% of users (i.e., those with the least incentive to contribute).

This motivates the main point of the present work: to make incentives more effective at encouraging contributions. The typical popularity of a single swarm is shaped like a bell curve with a long right tail, as most users leave immediately

after downloading. Ideally, we would like users to be able instead to draw on all previous downloads for service. The problem with tit-for-tat in BitTorrent is that the incentives are applied only when users are actively downloading, and only in the context of a single swarm. The observation made is that peers should instead be rewarded for all contributions across all swarms.

A simple fix would be for peers to cache a local history of their interactions with other peers and to reward peers with whom they repeatedly interact who have contributed to them in the past. However, a study of peer interactions shows that repeat interactions are very rare. The proposed approach is to implement one-hop reputations with limited indirection. One hop is enough, since almost all peers are connected through a very small number of the most popular peers. The protocol involves key pairs as long-term IDs and intermediary nodes to maintain accounting information and provide signed verification receipts. Intermediaries are discovered by gossiping during connection setup, and in the default policy they are selected based on popularity. Intermediaries receive priority service, and thus peers have incentive to serve as one. In the evaluation of the protocol, it was shown that 97% of random peers shared an intermediary; on average 73% of intermediaries selected in a random interaction are selected again within 100 interactions.

The authors conclude by pointing out that for peer to peer to reach its full potential, persistent contribution incentives are necessary, and one-hop reputations leverage the popular minority of peers for this purpose. One questioner asked why anyone would ever want to send directly to peers, since traffic sent to an intermediary is far more valuable in terms of the reciprocation it will garner. The response was that peer traffic's value will be inflated in the induced reciprocation economy.

- ***Ostra: Leveraging Trust to Thwart Unwanted Communication***
  Alan Mislove and Ansley Post, Max Planck Institute for Software Systems and Rice University; Peter Druschel and Krishna P. Gummadi, Max Planck Institute for Software Systems

Alan motivated his talk and work by pointing out that since digital communication such as VoIP, email, and IM is so low-cost, it can easily be abused to send unwanted communications. This manifests itself in various forms including spam and mislabeled content on YouTube, and users are not easily held accountable for their actions since new IDs can be created for free. Previous approaches to solving this problem—such as filtering content, charging money for sending messages, or introducing strong IDs—all have shortcomings.

The authors present a solution to this problem called Ostra, based on an ancient system for transferring money in India called Hawala. The basic idea is to leverage existing offline social trust relationships, since these are expensive to create and maintain. Most communication systems have some sort of implicit or explicit social network, and Ostra assumes that links in this network are maintained by some trusted site. Recipients of messages classify messages (e.g., perhaps implicitly via deletion) as wanted or not. Messages between peers are sent directly, but a link to a peer is broken over time if that peer is on a path to a destination that is receiving unwanted messages. Each link has a credit balance, and the balance is adjusted in favor of a message recipient if the message was unwanted and vice versa otherwise. The credit balance is bounded with a certain range, and the process is iterated over intermediate peers in the event of no direct link between the sender and the recipient.

The system guarantees that no user can send more spam than the amount of spam he or she has received plus the lower bound on link credit times the number of in-links he or she has. Further, this holds for any subgraph, showing that collusion doesn't help attackers and neither does the creation of many Sybil identities. Credit is decayed by a fixed percentage daily to prevent a user from unfairly being blocked. The authors simulated Ostra using a social network taken from YouTube, as well as an email trace from the MPI, and found that even with 20% of users being attackers, only four spam messages were received by any good user per day. Further, very few messages were delayed from links reaching their credit limits. One person raised the point that the classifications may not always be black and white; for example, a message might be unimportant now but important later on. The response was that the system can work alongside other systems such as whitelisting and that finding the proper classification notions is a difficult issue.

- ***Detecting In-Flight Page Changes with Web Tripwires***
  Charles Reis, Steven D. Gribble, and Tadayoshi Kohno, University of Washington; Nicholas C. Weaver, International Computer Science Institute

Charles began his talk by discussing the recent phenomenon of ISPs injecting ads into the Web pages their users visit. The work discussed attempts to detect such in-flight changes to pages and measure them. The system is implemented as JavaScript code, which runs in the client's browser, finds changes in the HTML of the page, and reports them to the user and a central server. It works by fetching and rendering the original page while fetching the JavaScript code in the background from the page's server as well. This code contains a compressed version of the page's expected source code, and the JavaScript compares the two versions. In a study involving 50,000 unique IP addresses, 657 clients saw changes from client software, ISPs, firewalls, and malware.

Some of these changes inadvertently broke some pages by causing JavaScript errors or interfering with forum posting, and others introduced security vulnerabilities such as cross-site scripting vulnerabilities. A major concern with this problem is that it affects all Web pages—similar to a UNIX root exploit—and that the Web developers are powerless to fix the problem. Thus the authors caution users about

software such as client proxies, since they wield root-like power. Further, the Web Tripwires tool helped find vulnerabilities in such software, which have since been fixed.

A publisher of Web content could react in various ways to its pages being altered. First, it could simply use HTTPS to encrypt its pages. However, this is both costly and rigid in that it can't allow security checks or caching. Web Tripwires offers an alternative that allows publishers to easily and cheaply detect most changes, at the cost of somewhat lesser robustness to attacks. The performance of Web Tripwires is also much better than HTTPS, both in terms of latency and throughput. More information on Web Tripwires can be found at http://vancouver.cs.washington.edu.

- ***Phalanx: Withstanding Multimillion-Node Botnets***
  Colin Dixon, Thomas Anderson, and Arvind Krishnamurthy, University of Washington

Colin began with a list of major botnet attacks that have occurred in recent years, including the government of Estonia being shut down by an attack for three to four weeks. He then posed the question, "Why isn't the problem with botnets solved?" In one sense, it is solved for static content, in that we can simply replicate content and use large CDNs. A potential solution for dynamic content might involve replacing all routers in the Internet, but this is not feasible. The key ideas in the current work's solution involve tieing the fate of a server to a large part of the Internet in a way that is scalable and deployable in the current Internet.

The mechanisms in the solution include numerous hosts used as proxies to make packet filtering decisions, forwarding the unfiltered traffic to the server we wish to protect. The nodes are used as mailboxes and hold each packet while waiting for an explicit request from the server. Secure random multipathing is used to protect communication. Traffic is sent randomly among the mailboxes according to a shared secret, and thus the botnet can only take one link down while communication still continues. The mailboxes negotiate a secret at connection setup time and use a lightweight authenticator. This scheme necessitates a multipath congestion control algorithm.

The problem still remains that if attackers sent traffic to the server directly they could still bring it down. Thus a filtering ring is used to drop unrequested Web traffic and to allow only requested traffic to reach the server exactly once. This is implemented by installing blacklists and whitelists on the server's routers. The scheme so far still only protects established connections between a client and server. To initiate connections, the server sends the first packet requests. Access to these requests is mediated by computational puzzles or authentication tokens. The authors evaluated their system by simulating attacks on PlanetLab, with favorable results.

An article about Phalanx begins on page 22 of this issue.

WIRELESS

Summarized by Geoffrey Werner-Allen (werner@eecs.harvard.edu)

- ***Harnessing Exposed Terminals in Wireless Networks***
  Mythili Vutukuru, Kyle Jamieson, and Hari Balakrishnan, MIT Computer Science and Artificial Intelligence Laboratory

The high-level goal of a MAC protocol is to transmit as many packets as possible. Today, the dominant approach to MAC protocols is CSMA (Carrier-Sense Multiple Access). However, the problem with CSMA is that it prohibits many transmissions that would have succeeded, owing to its failure to address the exposed terminal problem. This is the case where, although transmissions might seem to the senders to conflict, the recipients are sufficiently separated that they would have been able to receive the packet correctly. Instead of simple heuristic approaches that attempt to generalize rules to each node in the face of fluctuating bandwidth and channel properties, this work attempts to use empirical evidence to determine when overlapping transmissions can proceed.

Identifying simultaneous transmissions that can proceed safely requires that each node maintain a conflict map that describes whether or not it can transmit safely to node X if it overhears node Y transmitting. The conflict map is built based on observation of the loss rates associated with transfers. Once they reach 50%, throughput would be higher if the transmissions were scheduled sequentially rather than in parallel, so this is the threshold for inclusion in the conflict map. ACKs and the backoff policy must also be adjusted in the face of concurrent transmissions. To allow the node to observe when transmissions conflict, the MAC layer must both be able to recover the node address from unsuccessful receptions, which is facilitated by its inclusion in both the packet header and trailer, and pass up the header before the rest of the packet, so that it can be accurately time-stamped.

A prototype implementation is tested to see whether it can produce no-CSMA behavior when the terminals are exposed and CSMA-like behavior when the terminals conflict. Indeed, experiments on a multi-node 802.11 testbed show that their prototype is able to improve performance overall by essentially acting like CSMA only when CSMA is actually needed.

Questions for the presenter included the choice of 50% as the cutoff point for inclusion in the conflict map, whether or not weighing the signal-to-noise ratio against the noise floor might allow a simpler approximation of this algorithm, and whether or not experiments in noisier environments had been performed. Ms. Vutukuru responded that performance is similar across a wide range of cutoff points near the middle (30% to 60%), and that more tests were needed in different environments to evaluate the impact of varying parameters not yet experimented with.

- ***Designing High Performance Enterprise Wi-Fi Networks***
  Rohan Murty, Harvard University; Jitendra Padhye, Ranveer Chandra, Alec Wolman, and Brian Zill, Microsoft Research

Murty began by stating that more and more wireless is being deployed in the enterprise and users are beginning to develop the same high-capacity expectations for wireless performance as they have for wired. However, currently deployed enterprise wireless networks have many limitations. Because of a phenomenon known as "rate anomaly," the performance of deployed access points is limited by their slowest client. DenseAP seeks to revisit some of the original assumptions surrounding enterprise wireless networks, specifically that the number of access points should be much lower than the number of clients. By deploying a large number of access points and carefully controlling client associations, load balancing, and channel usage, DenseAP seeks to deliver wired-like performance over wireless links.

The challenges this work faces are threefold. First, deciding which wireless access points a client should associate with (controlling association). Second, determining which channel each access point should be operating on (channel assignment). Finally, as clients enter and leave the network and their bandwidth demands change, it is likely that associations will need to be revisited to balance load among access points.

DenseAP controls client associations through a central server, which, when a client begins sending out probe requests, decides which access point is the best match for that client and only allows that access point to respond to the probe request. Association policy is dictated by the quality of the connection and the demand present on each access point. In general, available capacity is equal to the expected transmission rate times the free air time at that access point. To estimate the available capacity the authors use a mapping between RSSI and throughput driven by empirical observations. To estimate free air time they observe the queuing delay at each access point. Channel assignment between access points is done by simply assigning each new access point to the least-loaded channel. As clients move and their behavior changes, associations may need to be reevaluated. To do this, the central controller actively shifts load away from access points that are incurring high stress.

The testbed used for the experiments in the paper is a portion of a floor of a corporate office building. While this area was normally served by only one corporate wireless access point, during experiments up to 24 DenseAP nodes (or DAPs) were used to service up to 24 clients. The authors present results showing improvements in overall performance, as well as attempting to isolate the effects of channel assignment, DAP density, and their intelligent association policy.

During questions, one person wondered whether it would be possible to also allow clients to use multiple access points. Mr. Murty replied that although this would require changes to the client, which DenseAP avoids, it would be interesting if possible. Professor Birman from Cornell asked about what happens if the client associated with the particular DAP selected by the central controller fails. Mr. Murty replied that when the central controller observes such a failure it will choose a new DAP for the client to associate with.

An article about DenseAP begins on page 41 of this issue.

- ***FatVAP: Aggregating AP Backhaul Capacity to Maximize Throughput***
  Srikanth Kandula, Massachusetts Institute of Technology; Kate Ching-Ju Lin, National Taiwan University and Massachusetts Institute of Technology; Tural Badirkhanli and Dina Katabi, Massachusetts Institute of Technology

Mr. Kandula described FatVAP, which is designed to address several problems in current wireless 802.11 networks. The first is that the backhaul bandwidth capacity of a particular access point may be bottleneck limiting flows, meaning that there is spare bandwidth at the sender that could be used to send data through other access points. The second is that choosing access points based on proximity combined with a high density of clients leads to hotspots—overutilization of certain access points, leaving spare capacity at others that competing clients could be utilizing. Ideal performance can be obtained by aggregating all access points usable by a particular client or set of clients into one virtual access point, with wireless and backhaul bandwidth equal to the sum of its parts. However, this requires clients to be able to multiplex their connections across multiple access points, which is currently not possible. That said, their solution, once implemented on one or a set of clients, requires no changes to the access points themselves to increase client performance.

To determine how to divide time among APs, FatVAP must solve a scheduling problem. In general, if we have a set of access points, each with a different drain capacity e and available bandwidth w, then a client need not connect to that access point for more than e/w of its time, referred to as the useful fraction. This quantity subsumes link quality, contention, and backhaul capacity. As several examples given showed, no greedy solution for this scheduling problem exists, as the problem is equivalent to a bin-packing problem, with the bandwidth being the value and the time spent at each access point being the cost, bounded by the total time available.

This approach is difficult and presents many implementation challenges. First, to estimate wireless bandwidth, synchronous acks can be used to measure the queue drain rate on each access point; estimating backhaul bandwidth can be accomplished through observing back-to-back large packets. To allow reception from multiple hosts, FatVAP uses 802.11 power save mode to compel access points to cache packets for it while it rotates through others it is using. A large set of client-side changes are needed, includ-

ing allowing the kernel to rotate through multiple APs by spreading traffic through a number of different interfaces above the kernel level. "Soft-switch" between access points allows them to enable high-rate TCP through multiple access points on top of FatVAP.

In conclusion, the authors have shown that FatVAP can aggregate throughput, balance load, and adapt to changing network conditions. In questioning, one person was curious about why the authors focused on bandwidth while neglecting latency. Mr. Kandula replied that further experiments were necessary to assess the impact of FatVAP on latency.

- ***Efficiency Through Eavesdropping: Link-layer Packet Caching***
  Mikhail Afanasyev, University of California, San Diego; David G. Andersen, Carnegie Mellon University; Alex C. Snoeren, University of California, San Diego

In real networks, overhearing happens, meaning that even if a route from A to C normally passes through B, some of the time C may overhear the packet being transmitted from A to B directly. In this case, it is advantageous to avoid retransmitting the packet that C already has from B to C. This scenario can also lead to unnecessary acknowledgment messages. Earlier solutions to the overhearing and multiple transmission problem have used caching, which introduces an unacceptable amount of delay at each client between transmissions.

To reduce retransmissions without introducing latency, RTS-id embeds a packet identifier in the RTS/CTS 802.11 exchange. The packet IDs are based on a hash of the packet contents, although RTS-id is careful not to include portions of the packet that may change as it traverses multiple hops.

RTS-id was implemented on top of Cal Radio, using packet modifications designed to look normal on nonparticipating nodes. The testbed consisted of three Cal Radio nodes, although simulations were also performed on data gathered from the RoofNet outdoor testbed. A state machine was used to model packet forwarding behavior during the simulations. Results show that RTS-id reduces retransmissions in the face of overhearing, with savings naturally scaling with the number of hops that the packet traverses. Because of the way that RTS-id was implemented it can also work seamlessly alongside nodes not implementing the protocol.

Professor Levis from Stanford was curious about whether the authors had investigated possibilities for spatial use as a result of their work. Mr. Afanasyev replied that they were considering this. Professor Karp from CMU asked whether or not this could be combined with other forms of network coding. Mr. Afanasyev wasn't sure.

LARGE SCALE SYSTEMS

Summarized by Jeff Terrace (jterrace@cs.princeton.edu)

- ***Beyond Pilots: Keeping Rural Wireless Networks Alive***
  Sonesh Surana, Rabin Patra, and Sergiu Nedevschi, University of California, Berkeley; Manuel Ramos, University of the Philippines; Lakshminarayanan Subramanian, New York University; Yahel Ben-David, AirJaldi, Dharamsala, India; Eric Brewer, University of California, Berkeley, and Intel Research, Berkeley

There has been considerable research done on deploying network infrastructure into developing, rural areas around the world, but the problem that Sonesh Surana et al. were trying to solve is that, once an infrastructure is in place, it's very difficult to keep it maintained and sustainable over long periods of time. Two existing wireless networks were studied: the Aravind Eye Hospital's video-conferencing network in southern India and the AirJaldi network in northern India, which provides Internet access to rural users. The largest problems facing sustainability are poor-quality grid power, limited local expertise for maintenance and diagnosis, lack of full connectivity in the network for remote management, and the physical location of networks residing in remote locations that are difficult to reach.

Hardware faults in these two networks were dominated by power-related faults. The problem in developing countries is that instead of a steady, reliable voltage rating, grid power can result in a large range of voltages, which ends up damaging electronic equipment. A UPS does not help, because although it provides reliable power during an outage, it passes power directly to the device during normal operation, which still results in bad voltages. Because commercial products were too expensive and sensitive, one solution was a custom-built low-voltage disconnect circuit to guard against low-voltage situations combined with solar power to handle peaks and swells in the power grid. A push-based PhoneHome system was also implemented that uses the cell phone network to report node, link, and network properties every 3 hours to a central server. Satellite links were also used to provide additional entry points into the network to address software and link failures causing some nodes to be unreachable. A cheap hardware watchdog device was also used to reboot routers that fail.

The additional devices and methods used here eliminated the need for weekly reboots, reduced power failures, and reduced prolonged downtime in the two networks; as a result, both networks are now financially stable.

- ***UsenetDHT: A Low-Overhead Design for Usenet***
  Emil Sit, Robert Morris, and M. Frans Kaashoek, MIT CSAIL

Emil Sit began by stating that there are over 2 million articles and files that arrive on Usenet every day, which translates to 30 MB/s. Usenet was one of the first P2P systems created. Servers that store Usenet articles are distributed geographically and as an article is posted to a server, the article is passed to the server's peers until eventually

the article is held on all Usenet servers. This system makes it difficult to create a Usenet server because of the large volume of data that the server must be able to store. UsenetDHT is a shared Usenet server that allows multiple servers to cooperatively share Usenet articles in a DHT.

UsenetDHT combines the storage space of multiple servers by distributing a single copy of a Usenet article among them. It separates article headers from article contents, and it stores only a single copy of an article's contents in the DHT. All servers keep a copy of the article headers (which makes up less than one percent of the storage cost) to allow clients to see headers immediately. By leveraging UsenetDHT, several small sites can benefit from the resources of its peers and can cooperatively run a Usenet server. UsenetDHT requires high throughput and data durability, but current algorithms for DHTs are synchronization heavy. Each node in the network must sync with several other nodes to provide durability and replication, which is a slow process over a WAN. To solve these problems, UsenetDHT uses Passing Tone, an algorithm on top of DHash that balances minimizing the bandwidth used between nodes and minimizing the amount of state that needs to be stored on each server. Passing Tone only keeps local synchronization data, shares the responsibility of ensuring proper replication with its neighbors, and can make decisions about replicas by only communicating with its immediate neighbors. Passing Tone is a simple algorithm that minimizes overhead but still performs almost as well as previous algorithms.

A question was asked about what UsenetDHT provides for censorship resilience. In reply, the speaker stated that there is an advantage in replicas being distributed, but that he doesn't envision UsenetDHT replacing Usenet because the latency might be too high across servers. UsenetDHT does not affect censorship. In reply to another question, the speaker stated that there is no public information about how much of Usenet is spam.

- ***San Fermin: Aggregating Large Data Sets Using a Binomial Swap Forest***
  Justin Cappos and John H. Hartman, University of Arizona

Justin Cappos said that computing results of a computation over a large, distributed data set can be difficult. When the amount of data you need to process is large, aggregating the data at a single location can take too long to process, so distributed aggregation algorithms have been devised. An example when this type of algorithm is needed is when a programmer is trying to analyze end-user traces of a program's execution. The programmer is only interested in a total sum, not individual values, but trace files can be too large to transfer to a single point. Instead, the traces can be processed locally and just the aggregated sums can be transferred as the result. The goals of an algorithm for aggregating large data sets are to have complete coverage, have no duplicates in the answer, have no partial data, tolerate node failures, not overload any individual node, and produce a result fast.

San Fermin is an algorithm for large data aggregation that uses a binomial swap forest to calculate results. Each node is assigned a unique identifier to prevent duplicate and missed nodes. Each node then starts swapping data with other nodes by considering each bit in its ID value from right to left and choosing another node that has a different target bit and has the same prefix. Each node has its own view of the network, that is, as a binomial swap tree, and by aggregating data with the nodes it chooses to swap with, it will eventually have the aggregated result that is desired. Since every node runs the algorithm, the first node to complete the aggregation reports its result and all other nodes can stop. This method allows the aggregation to be robust to node failures since a node will usually swap its data with multiple other nodes before failing, so its information is already in the binomial swap tree of others. The prototype of San Fermin is built on top of Pastry, which provides IDs, failure detection, and routing. To evaluate San Fermin, the prototype was tested on 100 PlanetLab nodes and compared to SDIMS, which is also built on top of Pastry.

The evaluation showed that both algorithms perform well with small amounts of data, but SDIMS starts to fall apart with a large amount of data. San Fermin has a much lower variance of completion time than SDIMS, and it scales much better. Increasing the number of nodes from 32 to 1024 or the data size from 256 kB to 1 MB only increases the completion time by a factor of 4. When 50 failures occurred during aggregation, the final results were only missing 5–11 nodes. If there is a large variance in bandwidth capacity across nodes, the faster nodes tend to finish first, which is a desirable property.

The conclusion was that San Fermin performs aggregation better than previous algorithms for large data sets, scales well, and is robust to failures. In reply to questions, Cappos stated that San Fermin applies only when exact results are needed instead of trying to approximate, and that San Fermin differs from MapReduce because MapReduce focuses on distributed computation, whereas San Fermin may only be computing a very simple operation but wants to avoid centralizing the data.

FAULT TOLERANCE

Summarized by Petr Marchenko (p.marchenko@ucl.ac.uk)

Awarded Best Paper!

- ***Remus: High Availability via Asynchronous Virtual Machine Replication***
  Brendan Cully, Geoffrey Lefebvre, Dutch Meyer, Mike Feeley, and Norm Hutchinson, University of British Columbia; Andrew Warfield, University of British Columbia and Citrix Systems, Inc.

Brendan Cully presented Remus, a system that allows unmodified software to be protected from the failures of the physical machine on which it runs. In case of a failure, a running system can continue its execution on an alternative physical host with only seconds of downtime while com-

pletely preserving its internal state. Remus uses virtualization, whereby protected software is encapsulated in a virtual machine, and its runtime state is propagated to a backup host at a high frequency, e.g., 40 times per second.

This state propagation is possible because of virtualization, which allows running VMs to migrate between physical hosts. Remus applies asynchronous whole-system replication at particular checkpoints, and the primary server remains productive between the checkpoints. Remus buffers output until a more convenient later time in order to delay the synchronization and perform payload computation. This technique is called speculative execution. It yields substantial performance benefits and allows checkpointing intervals on the order of tens of milliseconds. To keep the primary system and backup hosts consistent, Remus does checkpointing, which has to deal with CPU and memory replication, network buffering, and disk buffering. Memory and CPU replication are based on Xen's existing live migration mechanism. Network input and disk reads are applied to the system immediately; however, the network output and disk writes are buffered until a checkpoint is performed. Remus's protection overhead mainly exists from checkpointing and network delays introduced by network buffering.

Fred Douglis asked about the effects of network activity on high-throughput applications. Brendan said that the network-sensitive applications incur higher performance overhead. He added that the results of the SPECweb benchmark are presented in the paper and that Remus provides one-quarter of applications' native performance. Amin Bada questioned the large amount of data that was transmitted across the network, not all of which was strictly necessary for Remus's operation. Brendan acknowledged that they did not evaluate this in the work, but using more focused data might offer a significant performance improvement. Someone from the audience was interested in whether Brendan and his co-workers tried to experiment with hardware virtualization for their system. The speaker said that these sorts of experiments were not done because of the absence of appropriate equipment, but it would be an interesting direction for their future work.

- ***Nysiad: Practical Protocol Transformation to Tolerate Byzantine Failures***
  Chi Ho and Robbert van Renesse, Cornell University; Mark Bickford, ATC-NY; Danny Dolev, Hebrew University of Jerusalem

Distributed systems and protocols such as DNS, BGP, and OSPF are designed to tolerate only crash failures; however, it is crucial to have the ability to deal with Byzantine failures. Chi Ho discussed Nysiad, a technique for transforming a scalable distributed system or a network protocol designed to tolerate only crash failures to one that tolerates arbitrary failures. It uses a variant of Replicated State Machine (RSM) to translate Byzantine faults into crash faults.

The state machine of a host is replicated onto the guards of the host, together constituting an RSM. Nysiad's replication protocol, OARcast, ensures that the guards of the host remain synchronized. OARcast provides the following properties: All correct guards deliver a message if one correct guard does; the messages from a single origin are delivered in the same order; and a compromised host cannot forge a message of a correct host. When the communication graph is unknown, the common case, Nysiad has no good way of determining which hosts will be communicating with other hosts. In this case, the replication protocol will not work, as it relies on the trustworthiness of the sender's guards. The same problem arises when a host changes its guards or when reconfiguration takes place. To handle this problem, Nysiad introduces a logically centralized trusted certification service, Olympus. It is involved only when changing the communication and guard graphs. It produces signed certificates for hosts containing information that is sufficient for a receiver of a message to check its validity. Owing to the increased number of control messages sent per single end-to-end message, Nysiad's message latency is three times higher than the latency in the nonconverted system.

An attendee from Microsoft Research was curious how the system behaves when a host lies consistently. In response, Chi said that Nysiad includes additional protocols, which were not mentioned in the talk, that deal with this problem. An attestation protocol guarantees that messages delivered to the guards are a valid execution of the protocol and a credit protocol forces a host to either process all its input fairly or to ignore all input.

- ***BFT Protocols Under Fire***
  Atul Singh, Max Planck Institute for Software Systems and Rice University; Tathagata Das, IIT Kharagpur; Petros Maniatis, Intel Research Berkeley; Peter Druschel, Max Planck Institute for Software Systems; Timothy Roscoe, ETH Zürich

Byzantine Fault Tolerant (BFT) protocols for replicated systems have received considerable attention in the systems research community. However, it is hard to evaluate these protocols and distinguish the best one under certain conditions. This is because the BFT protocols are implemented in different languages, may require nontrivial libraries, and depend on particular systems. Thus, the implementation-based approach for comparison of BFT protocols is not always possible. Atul Singh presented BFTSim, a simulation environment for performance-modeling-based comparison of BFT protocols. The system includes a high-level protocol specification language, an execution environment, and a network simulator.

The protocol specification language allows one to capture the salient points of protocols without drowning in the implementation details (e.g., threads and cryptographic primitives). The network simulator provides the ability to explore protocols under different network conditions. The execution system runs the protocols and it emulates the execution overhead by introducing delays. Thus, a programmer has to specify the cost of a protocol's primitives, such as cryptographic operations.

Atul and his co-workers verified the correctness of BFTSim by comparing the evaluation of Zyzzyva, PBFT, and Q/U protocols under their simulator and the real evaluation presented in the literature. BFTSim was able to match the performance graph for the real protocols with an error less than 10%. BFTSim makes BFT protocols more accessible, as it offers a unified system for protocol performance comparison under certain network conditions.

There was a question about whether the protocol specification language captures the complexity of the protocol implementation such as lines of code. Atul explained that it does, as the amount of code in the specification language is proportional to the amount of code in the protocol's implementation.

MONITORING AND MEASUREMENT

Summarized by Eric Hielscher (hielscher@cs.nyu.edu)

- ***Uncovering Performance Differences Among Backbone ISPs with Netdiff***
  Ratul Mahajan and Ming Zhang, Microsoft Research; Lindsey Poole and Vivek Pai, Princeton University

Ming began by pointing out that there have been numerous studies done on evaluating and comparing systems such as file systems, databases, and Web servers but there has been little such work done on evaluating and comparing different ISPs. Thus customers don't have enough information to make a good decision as to which ISP is the best for their needs. The current state of the art involves service level agreements between customers and their ISPs in which the ISP guarantees some aggregate performance, something that doesn't easily translate into an assessment of perceived end-user experience.

The requirements the authors outlined to structure their study include that the ISP comparisons be both relevant to customers (by measuring end-to-end paths target destinations of interest and making comparisons based on workloads similar to their own) and useful to ISPs (by helping them to account for geographic presence and to identify bad Points of Presence [PoPs] or destinations). The ideal architecture of the comparison framework would involve deploying probes inside every PoP of the ISPs, and taking a probe from every PoP to every destination on the Internet. However, the overhead would be too high. In the Netdiff, probers are deployed at the edge of the network, and probes are sent from ends to various destinations on the Internet—for example, from an ingress PoP to an egress PoP of an ISP. A single centralized controller sends probe lists to all probers, which then send back their results.

The system is able to generate a complete snapshot of each of 18 backbone ISPs using between 5,000 and 23,000 probes in under 20 minutes, a significant improvement over another similar system called Keynote. The system is deployed on PlanetLab and has been generating such snapshots every 20 minutes for the past year. The comparison methodology involved using path stretch and grouping paths based on length and differentiating between paths to the destination on the Internet as well as internal paths, with ISPs ranking very differently on the various metrics. Detailed information on the data as well as the ability to generate individualized comparisons is available at http://netdiff.org.

- ***Effective Diagnosis of Routing Disruptions from End Systems***
  Ying Zhang and Z. Morley Mao, University of Michigan; Ming Zhang, Microsoft Research

Ying began by stating that the goal of his work is to diagnose routing disruptions purely by using end systems, a departure from existing approaches, since they are controlled by end users and needn't use ISPs' proprietary data. The desire for such diagnosis comes from the fact that such disruptions impact application performance as well as causing high loss and long delays. The approach taken only requires probing from end hosts with traceroute and can cover all PoPs of a target ISP as well as most destinations on the Internet. Disruptions are identified by comparing paths that are consecutively measured. Some challenges involved in this approach include limited probing resources, limited coverage of probed paths, and issues related to timing granularity and measurement noise.

The system's architecture involves collaborative probing by a set of distributed hosts, each of which sends traceroutes to different destinations on the Internet to learn routing state, improve coverage, and reduce overhead. Events are then classified according to ingress/egress changes into three types: the ingress PoP changes, the egress PoP changes, or neither does. Events are then correlated both spatially and temporally, since events happening close together in space or time are likely due to a few root causes. They employ an inference methodology by compiling pieces of evidence that support various causes such as an egress link being down. They then list all likely causes of each event of interest and build an evidence graph that maps evidence nodes to cause nodes close together in time. A conflict graph is also generated, with nodes that represent evidence that conflicts with a given event, to reduce cause candidate sets, and a greedy algorithm is used to search for a minimum set of causes while covering all evidence and having minimal conflicts.

Five large ISPs were monitored via a deployment on PlanetLab, covering all of the ISPs' PoPs, with refreshes occurring every 18 minutes. The results show that many events discovered were internal changes, something that BGP-based methods wouldn't find. The system was validated against existing BGP-based approaches as well, with somewhat high error rates owing to limited coverage, coarse-grained probing, and measurement noise. The system performed well enough to be usable for real-time diagnosis.

- ***Csamp: A System for Network-Wide Flow Monitoring***
  Vyas Sekar, Carnegie Mellon University; Michael K. Reiter, University of North Carolina, Chapel Hill; Walter Willinger, AT&T Labs—Research; Hui Zhang, Carnegie Mellon University; Ramana Rao Kompella, Purdue University; David G. Anderson, Carnegie Mellon University

Vyas began his talk by pointing out that the needs for network monitoring stem from things such as traffic monitoring, analysis of new user applications, and network forensics. In particular, good traffic measurements, including measurements of fine-grained traffic structure, are needed. Some design goals include limited resource consumption on routers, high flow coverage, ability to specify network-wide goals, and low data management overhead. Current systems for network monitoring employ uniform packet sampling on routers, with aggregation of individual router data into flow reports. This results in a bias toward large flows, coarse goal specifications, and redundant measurements.

The proposed system, cSAMP, randomly samples flows rather than packets to solve these issues. Each router hashes a tuple consisting of the network protocol and the source and destination IP addresses and ports to compute a FlowID. Each router in the network is configured to store a different subset of the hash function's range. This allows for global configuration (i.e., routers are not required to communicate during sampling). To allow for network-wide configuration, different hash ranges are configured per origin-destination pair in the network (e.g., NYC/PIT). A framework is provided for generating sampling manifests (the configuration files for the routers). This involves a linear programming problem, which takes as inputs origin-destination pair information and router resource constraints and outputs the optimal sampling strategy that maximizes traffic and coverage.

cSAMP was evaluated against fixed rate and maximal flow sampling as well as packet sampling. Its flow coverage was 2–3 times better than packet sampling and 30% better than maximal flow sampling. In addition, cSAMP is significantly better than the other methods at achieving minimal fractional coverage and network-wide goals. It is robust to traffic dynamics and scalable. A question was asked about whether cSAMP could be used for per-flow rate-limiting applications, and the response was that the current infrastructure is geared toward near-real-time analysis of flow reports rather than real-time monitoring for rate-limiting.

- ***Studying Black Holes on the Internet with Hubble***
  Ethan Katz-Bassett, Harsha V. Madhyastha, John P. John, and Arvind Krishnamurthy, University of Washington; David Wetherall, University of Washington and Intel Research; Thomas Anderson, University of Washington

Ethan started off his talk by pointing out that global reachability is a basic Internet goal. In use, the Internet seems usually to have such reachability, but there are numerous cases where this isn't true and there are transient reachability problems. The Hubble system aims to automatically identify persistent reachability problems. The algorithm it employs includes three steps. First, distributed ping monitors detect when a destination becomes unreachable. Second, reachability analysis is conducted using distributed traceroutes. Finally, the problem is classified. To detect whether the problem involves the forward or backward link between the source and the destination, Hubble employs IP address spoofing by having another source send a packet to the destination with the first source as its spoofed IP address. If the original source then hears a response, we can conclude that the problem was with the forward link; otherwise, it must be with the backward link.

Problems are detected by pinging destinations every two minutes. A destination is reported after a series of failed pings. A BGP table is maintained from RouteViews feeds, allowing for an IP-address-to-AS mapping. Next, the extent of the problem is assessed by using traceroutes to gather topological data, with probing continuing while the problem persists. Analysis is performed to determine which traceroutes reach the destination. Next, the problem is classified according to ISPs, routers, and destinations, in order to help operators diagnose and repair it.

The evaluation of Hubble presented in the talk focuses on two questions: How much of the Internet is monitored, and what percentage of paths is analyzed for each given prefix? The results show that, every two minutes, 89% of the Internet's edge space and 92% of ASes are monitored. In addition, for 60% of prefixes, Hubble monitored routes through all ASes on RIPE BGP paths to the prefix. Further results show that spoofing works well and that many Internet holes last for more than 10 hours and most were cases of partial reachability. An interesting result was that multihoming may not give resiliance to failure, since many multihomed prefixes had problems in which multiple traceroutes terminated in one provider while the prefix remained reachable through another provider. Hubble is running continuously, and a map of ongoing problems is available at http://hubble.cs.washington.edu.

PERFORMANCE

Summarized by Petr Marchenko (p.marchenko@ucl.ac.uk)

- ***Maelstrom: Transparent Error Correction for Lambda Networks***
  Mahesh Balakrishnan, Tudor Marian, Ken Birman, Hakim Weatherspoon, and Einar Vollset, Cornell University

Mahesh Balakrishnan started by explaining the problem that TCP/IP has when it is used in high-speed lambda networks. TCP/IP was designed to provide connectivity in congested networks; however, in networks such as TeraGrid with 40-Gbps links, there is no congestion but there are packet drops because of dirty fiber, misconfiguration, and switching contention. TCP/IP uses a feedback loop to recover lost packets, which results in dramatic throughput

reduction. A loss rate of 0.1% is sufficient to reduce TCP/IP's throughput by an order of magnitude.

As a solution for this problem, Mahesh proposed the Maelstrom Error Correction appliance, a rack of proxies residing between a sender and a receiver in a WAN link. These proxies apply Forward Error Correction (FEC) for the traffic being transmitted over the link. The sender proxy encodes every five data packets in three FEC packets, and the receiver proxy checks the correctness of data packets and uses FEC to recover lost or damaged data packets. This technique works well for random losses but not for burst losses. Therefore, Maelstrom uses a new encoding scheme called layer interleaving, which applies extra correction packets over the blocks of packets (layers), using three layers of different length.

Maelstrom was evaluated on the Emulab testbed. It is able to cope with loss rates up to 2% without significant throughput degradation, whereas TCP/IP's performance degrades dramatically when loss rate is increased from 0.01% to 1%. The overhead introduced by Maelstrom does not depend on the length of the links, but only on the data rate. The proposed solution is transparent, as it does not require modification of network infrastructure and software. Thus, TCP/IP can be run over Maelstrom.

Michael Walfish wondered why they rejected the possibility of using rateless code in their paper. Mahesh admitted that rateless code could be used in their system; however, he did not explain why they found it unusable but suggested taking this question offline. One attendee asked for a clarification of the difference between this work and the work presented at NSDI four years ago. Mahesh agreed that the work has some similarities, but the earlier one was doing error correction in the network; thus, their deployment models are completely different. Bob Read from Facebook asked whether Maelstrom supports $n + 1$ connectivity or whether there is always one-to-one mapping. Mahesh responded that if there are several connection points, they have to be paired; therefore, it is always one-to-one mapping.

- ***Swift: A Fast Dynamic Packet Filter***
  Zhenyu Wu, Mengjun Xie, and Haining Wang, The College of William and Mary

Zhenyu Wu addressed the problem of fast dynamic packet filtering. Dynamic filtering is essential for building network services, network engineering, and intrusion detection, where it is required to adjust the filter at runtime. When there is a dynamic filter update, the traditional filters such as BSD Packet Filter (BPF) requires three preprocessing phases: compilation of a new filter, user-kernel copying (as the filter runs in the kernel), and security checking to make sure that a new filter can be safely run in the kernel. These stages prolong filter update latency, which results in misses of hundreds or even thousands of packets. This gap can cause serious problems for critical applications such as intrusion detection systems.

Zhenyu presented SWIFT, a packet filter that takes an alternative approach to achieving high performance, especially for dynamic filtering tasks. Like BPF, SWIFT is based on a fixed set of instructions executed by the in-kernel interpreter. However, SWIFT is designed to optimize the filtering performance with powerful instructions and a simplified computational model. Powerful instructions allow SWIFT to accomplish common filtering tasks with a smaller number of instructions. This speeds up static filtering and allows removing the filter compilation stage in filter updates, which improves the dynamic filtering performance. SWIFT eliminates security checking during filter update; instead, it is banned from controlling the execution path and storing data. This prevents it from tampering with the kernel.

Simplifying the filter update procedure by removing the compilation and security checks allows SWIFT to achieve at least three orders of magnitude lower filter update latency in comparison with Linux Socket Filter (LSF). This reduces the number of missing packets per connection by about two orders of magnitude. The powerful instruction set and simplified computational model increase filtering speed; thus, SWIFT outperforms LSF by up to three times in terms of packet processing speed.

SECURITY

Summarized by Brendan Cully (brendan@cs.ubc.ca)

- ***Securing Distributed Systems with Information Flow Control***
  Nickolai Zeldovich, Silas Boyd-Wickizer, and David Mazières, Stanford University

It is very hard to build secure distributed systems. One major reason is simply code size: Application code can run into millions of lines, much of which is unaudited third-party library code of uncertain provenance. Within these large applications, even tiny vulnerabilities can lead to catastrophic data exposure. Although it isn't feasible to fix every application bug, systems such as Asbestos, HiStar, and Flume demonstrate that it is possible to prevent untrusted code from seeing private data in the first place. They do this using decentralized information flow control (DIFC) to track data as it flows across applications and enforce access control rules on that data. For example, a database credit card query might be labeled according to the user credentials supplied with it, and data flow control could then ensure that the response is only visible to the same application path that provided the credentials.

Current DIFC systems are limited to applications running on a single host. Nickolai Zeldovich presented a system, called DStar, that allows DIFC to be enforced across a network of mutually distrusting applications. This is done by

delegating local labels to an export process on each physical host, which uses self-signed labels (in which the public key of the exporter is part of the label name) to transfer labels over the network, where they may be converted back to local labels. To support decentralized flow control, any process can create new labels, remove labels it owns, and grant the ability to remove labels to other processes.

Because DStar's trust model is decentralized, it is possible to use flow control even across multiple operating systems. For instance, highly sensitive data might be processed under the HiStar environment, but less sensitive data could be handed off to Linux systems or even completely untrusted cloud computing systems.

An audience member asked how this category system differs from a normal capability system. Nickolai responded that categories are strictly more general. For instance, they make it possible to assert negative access rules.

- ***Wedge: Splitting Applications into Reduced-Privilege Compartments***
  Andrea Bittau, Petr Marchenko, Mark Handley, and Brad Karp, University College London

The number of reported security vulnerabilities continues to increase every year. This is partly because programmers ignore the principle of least privilege. Andrea Birtau argued that they do this because the process-based privilege mechanisms commonly available now are not fine-grained enough to provide good protection. Wedge is a system designed to help with this problem in two ways: first, by providing a simple partitioning mechanism called sthreads, which only shares memory with other sthreads that have been explicitly tagged, and second, by introducing a tool called crowbar which helps to find good partition strategies for existing legacy applications.

The standard approach to privilege separation is to fork when changing privilege levels, using file descriptors between processes to share data. Unfortunately, it's easy to accidentally leak information to child processes, because memory is copied into them by default, and so it must be manually scrubbed before the child begins executing. It's also hard to share information, because it must be serialized across a file descriptor. Sthreads avoid the latter problem because, like standard threads, they run in the same address space. They also avoid the former problem because they can only see memory in other sthreads for which they have been granted a capability. Sthreads associate a tag with all memory they allocate and may grant tags to other sthreads.

It would be desirable to apply this mechanism to existing code, but ad hoc analysis of how data is shared among tasks in an existing monolithic application is impractical. For example, the Apache Web server accesses over 600 different memory objects. Manually tagging each of them would be both painful and hard to get exactly right. Static analysis may also fail (e.g., because of function pointer usage). Crowbar attempts to discover partitions by observing the actual access patterns of running applications, using the Pin dynamic instrumentation system.

Andrea was asked about his experience designing applications using Wedge. He told the audience that he wrote a DNS server from scratch with sthreads and didn't feel that explicit memory tagging required significant extra effort. Converting legacy code was of course much more difficult.

ENERGY

Summarized by Geoffrey Werner-Allen (werner@eecs.harvard.edu)

- ***Reducing Network Energy Consumption via Sleeping and Rate-Adaptation***
  Sergiu Nedevschi and Lucian Popa, University of California, Berkeley, and Intel Research, Berkeley; Gianluca Iannaccone and Sylvia Ratnasamy, Intel Research, Berkeley; David Wetherall, University of Washington and Intel Research, Seattle

The rising energy consumption of networking-related equipment is a pressing issue, given rising energy costs and the increased recognition of the impact of CO_2 emissions on the global climate. Given that most network equipment is provisioned for maximum load, which is rarely reached, network devices provide a great opportunity for power savings. Power consumption should reflect utilization, not capacity. This work explores two techniques to reduce power consumption: sleeping, that is, disabling routers for periods of time, and frequency scaling, that is, reducing the processing speed of the router itself. Combined, these techniques should reduce both the active and the idle power consumption of routers. Given that new routers are beginning to be shipped with the ability to sleep and rate-adapt, these techniques are a promising way to reduce power consumption while protecting performance.

In general, sleep states consume much less power than even idle ones; however, the transition time to awaken the router when data arrives is a concern. The authors assume that the router can be awakened by either a timer or link activity. To create periods in which a link can sleep, they buffer packets at the link and then transmit them through in a burst. Coordinating these buffer and burst periods throughout a network can ensure that the introduced latency does not increase as data traverses multiple hops. However, it turns out that their experiments show that the benefits of coordinating sleeping are minimal compared to uncoordinated sleeping, which captures most of the available energy savings.

Rate adaptation involves lowering the processing rate of the router to just the point necessary to keep up with link traffic. In general, the perfect link adaptation algorithm is not implementable, as it requires future knowledge of link activity. Instead, the heuristic algorithm the authors implement observes the local queue depth and current rates in order to choose future rates. As their analysis shows, uniformly

spaced variable rates are better for capturing the benefits of rate adaptation then the exponentially spaced rates available on many routers currently shipping. The authors evaluated their approach through simulations using power states and transitions from an Intel NIC. They found that, on this particular card, sleeping produced better results than rate adaptation, but they point out that this card, like many others, was not really designed for power savings. They then present a complete model of power savings that will allow them to evaluate future cards. Finally, they showed results indicating that, when comparing rate adaptation and sleeping, there is a utilization threshold that serves as a crossover point: Under it, sleeping performs better; after it, rate adaptation performs better.

Brian Zill from Microsoft Research asked whether the hardware used for this purpose in typical networking is optimized for energy consumption, and whether many of the benefits they described may be achieved simply through better hardware design. Mr. Nedevschi responded that today's networking hardware is indeed not particularly power-aware, but that this is beginning to change. Another questioner asked what percentage of the total energy consumption of networked equipment was tied up in the switching hardware that they are improving. Mr. Nedevschi wasn't sure. Finally, another questioner asked how this would affect TCP congestion control, and Mr. Nedevschi pointed out that they were careful that their changes produced no impact on TCP performance.

- ***Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services***
  Gong Chen, University of California, Los Angeles; Wenbo He, University of Illinois at Urbana-Champaign; Jie Liu and Suman Nath, Microsoft Research; Leonidas Rigas, Microsoft; Lin Xiao and Feng Zhao, Microsoft Research

Jie Liu stated that IT servers are the energy hog of the IT industry. The speaker pointed out that the increase in server energy consumption between 2000 and 2006 was enough to power 5.8 million American homes. Obviously, there is a chance to save a significant amount of energy if server energy consumption can be better managed. And the opportunity exists, because server load fluctuates for a variety of reasons throughout the day. This work focuses on adjusting the number of servers needed to serve MSN Messenger, a connection-intensive application. Because it is costly to migrate connections between servers, the authors focus on predictive techniques to identify the number of servers to have active at any given moment, combined with different approaches to load balancing connections across the servers active at any given moment. In addition, because shutting down servers conserves the most energy, the authors focus on ways to completely shut down servers when not in use.

A brief overview of the MSN Messenger architecture was presented. The servers targeted for power savings are the connection servers, which are in charge of maintaining persistent client connections but not storing a great deal of client state. The authors identify three metrics: service availability, service continuity, and service latency. Each server has a bounded number of connections and a bounded connection rate at which it can accept new connections. For the servers and application studied, the number of connections is on the order of 100k, whereas the server can only add around 70 new connections per second, meaning that they can be modeled by leaky buckets with tiny input pipes. The slow speed with which connections can be added also makes forward-looking provisioning all the more important.

The first step is load forecasting, in which regression models incorporating daily and seasonal fluctuations, along with the current state of the system, are used to predict load. In the experiments they performed, the system was trained on five weeks of data and then tried to predict a single week. Load dispatching is another key part of the system since the rates with which users can be added are limited and the distribution of users affects which machines can be shut down and how many users will be affected. The balancing approach assigns users to all available machines roughly evenly, whereas the skewing approach assigns users to fill one machine at a time.

Their evaluation looked at several different combinations of these approaches (e.g., skewing versus balancing and forecasting versus no forecasting). What they found is that skewing plus forecasting performs the best, with a 30% reduction in energy used. A number of graphs pictorially demonstrating the impact of different load balancing policies were shown. Finally, the authors identify a number of alternate approaches, including TCP state migration as well and building support into the client, allowing it to handle requests to move to a different server.

Rik Farrow of USENIX asked about some sharp spikes in the load graphs that had been shown. These turned out to be due to code rollouts or the effect of shutting down machines and having a bunch of clients reconnect all at once. Professor Vahdat from UCSD asked about using virtual machines to assist in the state migration, to which Mr. Liu responded that their servers don't actually maintain much state.

ROUTING

Summarized by Petr Marchenko (p.marchenko@ucl.ac.uk)

Awarded Best Paper!

- ***Consensus Routing: The Internet as a Distributed System***
  John P. John, Ethan Katz-Bassett, Arvind Krishnamurthy, and Thomas Anderson, University of Washington; Arun Venkataramani, University of Massachusetts Amherst

Internet protocols have traditionally favored responsiveness (a liveness property) over consistency (a safety property). Thus, they apply routing updates immediately to its forwarding tables before propagating them to other routers.

This causes routing loops and blackholes. Fully 10%–15% of BGP updates cause loops and 30% packet loss.

John P. John presented consensus routing, where he proposed to separate safety and liveness properties using two models of packet delivery: stable and transient. A stable mode ensures that a route is adopted only after all dependent routers have agreed upon a consistent view of the global state. This is achieved by a distributed snapshot and a consensus protocol. Transient mode ensures availability when a packet encounters a router that does not possess a stable route because of a link failure or an incompletion of consensus protocol. In this case, the router makes forwarding decisions based on transient heuristics such as backup routes, deflections, and detours. Consensus routing, which resides as a layer on top of BGP, does not require changes to BGP and does not disclose any more information regarding its routing policies than BGP does.

Comparison of BGP's connectivity and consensus routing connectivity in case of AS traffic engineering (prefix withdrawing) showed that BGP maintains connectivity only in 40% of the test cases, whereas consensus routing does so in 99% of the test cases. The loss of connectivity happens because of the transient loops. Consensus routing was able to converge from one consistent state to another, thereby avoiding transient loops in all test cases. Consensus routing adds traffic overhead, as it requires 30% more update bytes than BGP.

There was a question about whether a policy is more or less opaque with consensus routing than with BGP. John answered that since the entire policy is not known to all, it is as opaque as with BGP. Another question concerned whether it is possible to bundle updates. The answer was that bundling would cause inconsistency as a single update propagate policy that affects individual ASes. Someone wondered about the downtime in traffic forwarding that is caused by performing updates. John said that the effect of applying updates is covered by detour routing in the transient stage.

- ***Passport: Secure and Adoptable Source Authentication***
  Xin Liu, Ang Li, and Xiaowei Yang, University of California, Irvine; David Wetherall, Intel Research Seattle and University of Washington

Xin Liu addressed the problem of source address spoofing, since it damages the Internet in a variety of ways. Address spoofing significantly mitigates the effectiveness of DoS defense mechanisms. It also makes possible reflector attacks and makes source address filtering untrustworthy.

Xin Liu proposed Passport, a novel network-layer source authentication system. Passport treats an AS as a trusted and fate-sharing unit, and it authenticates the source of a packet to the granularity of the origin AS. It uses symmetric-key cryptography and checks packets only at administrative boundaries. When a packet leaves its source AS, the border router stamps one Message Authentication Code (MAC) for each AS on the path into the packet's Passport header. When the packet enters an AS on the path, the border router verifies the corresponding MAC value, using the secret key shared with the source AS. The correct MAC can only be produced by the source AS that also knows the key.

Passport relies on the routing system to efficiently manage keys using Diffie-Hellman key exchange on routing advertisements. Source address spoofing within a single AS is considered to be an internal issue for an AS. This solution can be incrementally deployed, as it is interoperable with legacy ASes.

An attendee asked about the routing assumptions. Xin stated that Passport requires routers having complete routing tables. Another question was asked about MTU because Passport adds MACs. Xin said that this is not an issue for routers; they can increase the size of the packet and they do it anyway for things such as VPNs.

- ***Context-based Routing: Technique, Applications, and Experience***
  Saumitra Das, Purdue University; Yunnan Wu and Ranveer Chandra, Microsoft Research, Redmond; Y. Charlie Hu, Purdue University

Saumitra Das discussed the effects of new lower-layer technologies such as multiple radios and link layer network coding on the routing path in wireless mesh networks. As he pointed out, conventional routing frameworks do not allow taking advantage of the new lower-layer technologies, since the costs of the links are examined in isolation from each other. Thus, multiple radios and network coding are not considered by conventional routing mechanisms.

Saumitra suggested a framework for routing in the presence of inherent link interdependencies, called context-based routing. It includes a new context-based path metric and route selection method that leverage the advantages of network coding and multiple radios. This context-based framework uses conditional link metrics: the Expected Resource Consumption (ERC), which models the cost saving from network coding, and a Self-Interference–aware Metric (SIM) for multiple radio systems. A context-based path pruning method uses these metrics to identify a preferable path. Based on these primitives, Saumitra and his colleagues implemented a Context Routing Protocol (CRP) and conducted experiments on two testbeds, demonstrating significant throughput gains.

An attendee asked whether the links advertised by CRP would already be congested. Saumitra said that the lower cost is advertised based on the throughput gain that you would get. One attendee wanted to know how much predictability you need in the flows to calculate the correct cost of the flow. The response was that they have a mechanism in the paper to ensure that equilibrium is reached.

UNDERSTANDING SYSTEMS

Summarized by Brendan Cully (brendan@cs.ubc.ca)

- ***NetComplex: A Complexity Metric for Networked System Designs***
  Byung-Gon Chun, ICSI; Sylvia Ratnasamy, Intel Research Berkeley; Eddie Kohler, University of California, Los Angeles

Simplicity has always been valued very highly in system design, but it is hard to measure quantitatively. Crude metrics like number of messages or total state size can be very misleading; for example, flooding is simple but produces many messages. Byung-Gon Chun presented a new metric, called NetComplex, to better reflect our intuition about the complexity of the algorithmic component of networked systems. It is based on the observation that these systems center on distributed state, and this state is dependent on the messages that communicate it. NetComplex uses a dependency graph in which discrete elements of single-host state form the nodes of the graph and messages that change that state form the edges.

NetComplex divides complexity into two levels. The most basic level is state complexity, which is the number of state changes that occur across the dependency graph as a result of changes to each variable. Operation complexity is a higher-level metric which aggregates the total state complexity resulting from an operation as defined by the system API. This is the metric by which Byung-Gon proposed that alternative algorithms be compared.

The rest of the presentation attempted to demonstrate the accuracy of the metric, first by using it on several different routing protocols, where it was determined that compact routing was the most complex protocol in spite of the fact that it had both the least state and the fewest messages (because it was designed for scalability). The metric was also applied to a number of classical distributed systems and then compared to the complexity rankings assigned by a survey of 19 graduate students in a distributed systems class; they matched closely.

There were a number of interesting questions. One attendee observed that Ethernet was a wildly successful algorithm, but according to this metric it would be classified as extremely complex (owing to exponential backoff). Another attendee pointed out that conventional metrics apply to resources, so that a system with limited bandwidth might optimize for fewer messages at the expense of more state. He wondered how NetComplex was intended to be used to select systems given that it did not apply to particular resources. Byung-Gon replied that, in general, the simplest algorithm was the best choice for producing robust systems.

- ***DieCast: Testing Distributed Systems with an Accurate Scale Model***
  Diwaker Gupta, Kashi V. Vishwanath, and Amin Vahdat, University of California, San Diego

A recurring problem for application developers is that they simply do not have the resources to test their applications in all of the different environments in which they will eventually be deployed. DieCast is a system that attempts to replicate large systems with a high degree of fidelity on a much smaller number of machines, while also providing reproducibility and making efficient use of the available hardware. Its approach is to use virtualization to multiplex many logical machines onto a single physical host, and then to carefully manipulate perceived time within the VMs to adjust for the reduced CPU available to them. This allows CPU to scale to large numbers of logical systems, but it does not scale either RAM or disk capacity.

Within a single virtual machine, time dilation (presented at NSDI '06 by the same group) can be used to hide increased runtime from the running operating system. But in such an environment, unmodified I/O would appear correspondingly faster. For example, in a VM in which virtual time progresses at one-tenth the speed of real time, a 1-Gbps network link would appear to run at 10 Gbps. Therefore, DieCast interposes on network and disk devices to scale them according to the time dilation factor in effect, so that perceived latency and throughput match those of the real devices.

The accuracy and utility of DieCast were evaluated in two ways. First, the RUBiS Web application benchmark was run natively on 40 nodes, and under DieCast on 4 nodes of 10 VMs each. The resulting throughput and response time scores matched very closely. A case study was also provided in which a scalable storage company reported good results from testing changes to their high-performance computing application.

- ***D^3S: Debugging Deployed Distributed Systems***
  Xuezheng Liu and Zhenyu Guo, Microsoft Research Asia; Xi Wang, Tsinghua University; Feibo Chen, Fudan University; Xiaochen Lian, Shanghai Jiaotong University; Jian Tang and Ming Wu, Microsoft Research Asia; M. Frans Kaashoek, MIT CSAIL; Zheng Zhang, Microsoft Research Asia

It is difficult to debug distributed systems, in particular because it is hard to reproduce error conditions. Machines run concurrently at varying speeds, and network conditions change dynamically. For example, a distributed lock manager may provide exclusive or shared locks with the invariant that only one client can hold an exclusive lock. Optimizations such as local state caching can make it tricky to reason about whether the invariant always holds. Simulation and model checking can help, but only to a degree. Eventually, runtime checking is likely to be necessary.

The most common approach to runtime checking is to add logging to an existing system and then to attempt to replay

from the logs. This can entail considerable developer effort, and getting just the right level of logging can require many iterations: Too much logging can produce unacceptable overhead, but too little will miss key state changes. And even after the logs are captured, analysis remains challenging. D^{3}S attempts to simplify the process of runtime assertion checking, by letting developers add distributed assertions to running systems on the fly. The primary contributions of D^{3}S are a simple language for distributed predicates, the ability to inject predicates into running systems, and tolerance of host or network failures. D^{3}S injects code into running systems by rewriting the running binary at specified hook points to collect assertions. These are sent to a set of assertion-checking servers using messages tagged with a Lamport clock to form globally consistent snapshots. In order to tolerate failure, each node provides a heartbeat, the loss of which removes it from the snapshot set.

The authors used D^{3}S on five real systems (all third-party applications) to evaluate whether it helped to find bugs. They found that it was easy to write predicates for these systems and that they were able to discover bugs that required runtime checking. Because only assertion state was logged and checked, the overhead on running systems was low (between 3% and 8%).

One audience member wondered how one could specify a predicate that could be used to find performance problems. Xuezheng acknowledged that this was a tricky problem, but argued that being able to add and remove probes on the fly would still be very helpful. Another attendee asked how probes could be written for applications written in higher-level languages other than C or C++. Xuezheng claimed that most real applications are written in C/C++ and that higher-level languages often provided better debugging facilities directly.