conference reports

THANKS TO OUR SUMMARIZERS

7th USENIX Symposium on Networked Systems Design and Implementation (NSDI '10)

San Jose, CA
April 28–30, 2010

OPENING REMARKS AND AWARDS PRESENTATION

Summarized by Rik Farrow (rik@usenix.org)

Alex Snoeren started the conference with the numbers: 224 attendees, 175 papers submitted, three rejected for not following formatting rules, and 29 accepted for presentation. He went on to thank the Program Committee, sponsors, and USENIX staff before announcing the Best Paper award.

The "Reverse Traceroute" paper (Ethan Katz-Bassett, University of Washington; Harsha V. Madhyastha, University of California, San Diego; Vijay Kumar Adhikari, University of Minnesota; Colin Scott, Justine Sherry, Peter van Wesep, Thomas Anderson, and Arvind Krishnamurthy, University of Washington) got the award. Snoeren then introduced Laura Hess, vice chair of the Computer Research Association (CRA), who explained that the CRA mission is to strengthen CS research as well as to support participation by women and minorities. Hess announced the female winner of the CRA, Justine Sherry (also one of the authors of the Best Paper award). Sherry has done a lot of work using timestamp algorithms in IP headers, has just graduated from University of Washington, and has been accepted into UC Berkeley.

CLOUD SERVICES

Summarized by Andrew D. Ferguson (adf@cs.brown.edu)

- ***Centrifuge: Integrated Lease Management and Partitioning for Cloud Services***

Atul Adya, Google; John Dunagan and Alec Wolman, Microsoft Research

Many distributed applications partition responsibility across multiple application servers and employ leases on those responsibilities in order to handle server churn. Alec Wolman spoke about Centrifuge, a reusable component which implements these services for Microsoft's Live Mesh platform. Centrifuge is targeted at interactive applications whose servers keep many small data objects in memory and operate on data cached from either the clients or back-end storage.

The Centrifuge architecture consists of three components: lookup nodes, which make requests for data objects; owner nodes, which have a lease on the cached data; and the Centrifuge manager service. The Centrifuge service hands out leases on ranges in order to scale to very large numbers of objects and uses consistent hashing to partition the namespace. To ensure consistency when handling an application-specific operation, an

owner checks that it owns the lease both before and after performing the operation; this is known as the "check-lease continuous" call. Leases are granted for 60 seconds and are renewed every 15 seconds.

When an application server crashes, systems using Centrifuge recover data from the client. This is cheaper than holding multiple copies in other servers and avoids the complexity and performance issues of quorum protocols. Wolman argued that this does not add extra complexity, because such recovery is already necessary for tolerating correlated failures. Centrifuge is currently designed for about 1000 servers in a single cluster.

Ben Reed asked if they had investigated lowering the lease time from 60 seconds, which he felt was a long time for an interactive system. Wolman responded that higher lease times lead to greater system stability and that the delay from re-fetching the data from stable storage (particularly if it is on the client) has a greater impact on system interactivity. Gün Sirer remarked that they are implementing load-link/store-conditional semantics and wondered what happens when the "check-lease continuous" call fails; he also asked if the operations needed to be side-effect free. Wolman confirmed that the owner nodes can't expose state and that if "check-lease continuous" failed, the server was expected to contact the lookup server in order to properly redirect the client's request.

- ***Volley: Automated Data Placement for Geo-Distributed Cloud Services***
  Sharad Agarwal, John Dunagan, Navendu Jain, Stefan Saroiu, and Alec Wolman, Microsoft Research; Harbinder Bhogan, University of Toronto

Cloud service providers use multiple data centers in order to reduce the latency when serving requests from globally distributed users. Of course, provisioning capacity at these data centers and WAN bandwidth between them are expensive. Sharad Agarwal presented Volley, an algorithm for partitioning data across data centers while minimizing end-user latencies and datacenter costs. Volley handles the data placement calculations for the Live Messenger service, which supports on the order of 100 million users conducting around 10 billion conversations per month.

Volley processes log data to produce a graph with connections between IP address locations and the data items accessed from those locations. There are also connections representing interactions between data items. All connections are weighted by how frequently the interactions occurred. The Volley algorithm then proceeds in three phases. In the first phase, a good initial solution is calculated by finding the weighted spherical mean of accesses to each data item. In the second phase, this solution is iteratively updated by using a weighted spring model which attempts to pull data closer to users. In the final phase, Volley assigns each data item to the data center closest to the optimal location while respecting capacity constraints.

After explaining the algorithm, Agarwal presented a comparison of Volley against alternate placement strategies for data in Microsoft's Live Mesh service. Volley reduced WAN traffic by 1.8x, reduced data center skew by 2x, and reduced user latency by 30% at the 75th percentile.

Chip Killian asked if there was a way to determine which latencies mattered the most to users. Agarwal replied that the services that use Volley do not currently collect that information. He mentioned that they had considered using the location information specified in users' account profiles, but found the listed location to match the use locations for only 50% of cases. Agarwal was also asked how they account for users traveling and he responded that they weight the importance of each access location by the number of requests the user made from that location. Ben Zhao asked about the role of CDNs in Volley. Agarwal said that Volley is designed for operators of dynamic services who don't want to pay the cost and overhead for the data to be replicated and consistent across the CDN.

- ***Optimizing Cost and Performance in Online Service Provider Networks***
  Zheng Zhang, Purdue University; Ming Zhang and Albert Greenberg, Microsoft Research; Y. Charlie Hu, Purdue University; Ratul Mahajan, Microsoft Research; Blaine Christian, Microsoft Corporation

Ming Zhang discussed the problem of traffic engineering in online service provider (OSP) networks, such as those run by Google, Microsoft, and Yahoo!. The goal of traffic engineering in an OSP network is to reduce the round-trip time experienced by end-user requests. This is a difficult objective, because OSP networks contain dozens of data centers and connect to hundreds of ISPs, and each ISP offers paths to only certain destination prefixes. Furthermore, path capacity constraints create dependencies among prefixes.

To solve this problem, a system called Entact was proposed. Entact injects /32 routes into the network in order to measure the performance of unused alternate paths to each prefix without affecting production traffic. These measurements are then used in a linear programming model which maps the cost/performance trade-off curve of a given traffic engineering strategy. The network operator is then able to select the strategy that makes the appropriate trade-off. Entact was applied to a simulation of Microsoft's OSP network and achieved a 40% cost reduction without degrading end-user performance.

Zhang was asked about the overhead of probing all of the prefixes to determine their performance. Entact only probes at a rate of about 1 Mbps from each data center, sending the probes from the two nearest data centers to each prefix, and it doesn't evaluate all alternative paths equally. Could Entact use actual traffic statistics for performance prediction instead of doing route injections? By only examining traffic, Entact might miss some alternative routes which could be evaluated by route injections.

WIRELESS 1

Summarized by Ann Kilzer (akilzer@cs.utexas.edu)

- ***Exploring Link Correlation for Efficient Flooding in Wireless Sensor Networks***
  Ting Zhu, Ziguo Zhong, Tian He, and Zhi-Li Zhang, University of Minnesota, Twin Cities

Ting Zhu explained how they created a flooding protocol that provides high reliability with fewer retransmissions. They began by challenging the common assumption of link independence in wireless networks. Through an empirical study conducted both indoors and outdoors, they found that packed receptions are in fact highly correlated. The authors leveraged this discovery to reduce redundancy and provide increased reliability by developing "collective acknowledgment" and using Dynamic Forwarder Selection, the details of which are in the paper.

Next, Zhu illustrated the Collective Flooding (CF) protocol and its difference from traditional flooding with a simple three-node network. In traditional flooding, a sender needs to wait for both receiving nodes to send ACKs. However, suppose that receiver 1 has a 100% conditional packet reception probability for receiver 2, i.e., the probability that receiver 2 has received the packet given that receiver 1 has received the packet is 100%. Then the sender only needs to get an ACK from receiver 1 to infer that both receivers got the message. Zhu then detailed the CF protocol and its three stages: maintenance, sender, and receiver. The nodes need different information at each stage. Nodes in the maintenance stage need link quality and conditional packet reception probability. Nodes in the sender and receiver stages need to know which nodes have not received the message (uncovered nodes), as well as coverage probabilities of neighboring nodes. A back-off timer is used to determine which node sends retransmissions. Throughout the protocol, nodes calculate coverage probabilities for all one-hop neighbors. When the coverage probability exceeds a threshold, the flooding protocol terminates.

To evaluate their flooding protocol, the authors used indoor and outdoor tests, as well as large-scale simulations. Additionally, they compared CF to other flooding algorithms such as RBP8 (a variation of RBP), RBP, and standard flooding. They found that CF reduced total transmissions by 31.2% and dissemination delay by 55% against RPB8. Additionally, CF had similar reliability to RBP8 and reduced the number of redundant retransmissions. Further information is available at www.cs.umn.edu/~tzhu.

Aaron Schulman from the University of Maryland asked why there was a correlation in packet reception between nodes. Zhu explained that background noise can cause correlation. Additionally, receivers have different sensitivity to packet receptions, so some nodes will always receive more packets. Philip Levis from Stanford University asked why there was a correlation in the outdoor context, because there should be fewer physical obstacles. Zhu again noted that this was because the receivers have different sensitivities to packets.

- ***Supporting Demanding Wireless Applications with Frequency-agile Radios***
  Lei Yang, University of California, Santa Barbara; Wei Hou, Tsinghua University; Lili Cao, Ben Y. Zhao, and Haitao Zheng, University of California, Santa Barbara

Lei Yang began by noting the recent growth of multimedia streaming in home and office environments, emphasizing the high bandwidth and quality-of-service requirements. Their work was motivated by a desire to provide streaming in a wireless environment. They aimed to build a system to provide continuous access to radio spectrum with high-bandwidth transmissions, even with concurrent traffic. An obvious solution would be to use WiFi; however, Yang noted that there is no dedicated access, and CSMA contention causes many unpredictable disruptions.

The authors proposed per-session frequency domain sharing, which enables concurrent media sessions on separated frequencies. This solution would avoid interference and allow continuous spectrum access for each media session. Yang noted that the FCC has auctioned and released new spectrum, making this work all the more feasible. He then introduced Jello, a decentralized home media system based on frequency domain sharing. This system uses frequency-agile radios, which sense and select free spectrum for media sessions.

Yang noted some of the interesting research and engineering problems in building the Jello system. First, there is the problem of finding free spectrum. Conventional methods use energy detection, which is unreliable. Yang demonstrated how to find free spectrum by using an edge-detection algorithm on a power spectrum density map. A second problem is choosing frequency blocks. Jello uses a best-fit approach, meaning that the radio will broadcast on the narrowest available frequency block that will sustain the current demand. A third challenge is fragmentation of the spectrum as links enter and leave. This can result in situations where there are enough bands to support a request, but they are not contiguous. Yang compared spectrum fragmentation to disk fragmentation, but noted that unlike on hard drives, a centralized defragmenter would not work because it would require stopping all transmissions. The authors proposed a solution of voluntary spectrum changes to lower fragmentation. They also proposed a second solution of non-contiguous spectrum access. However, this adds more overhead. Additionally, more fragmentation means more spectrum lost to guard bands, which are required between each link.

For evaluation, they used an 8-node GNU radio testbed with four concurrent flows. They evaluated a static system with equally partitioned spectrum, Jello with contiguous frequency access (Jello-C), Jello with non-contiguous frequency access (Jello-Full), and a simulated optimal solution without fragmentation or overhead. They measured the percentage of time a video stream was disrupted. In Jello-

Full, disruption was under 5%, while Jello-C had disruption around 18%. The static solution had 23% disruption. A full implementation of Jello can be found at http://www.cs.ucsb.edu/~htzheng/papyrus.

Sayandeep Sen from the University of Wisconsin-Madison wanted to know if the disruption metric referred to frame freezes or glitches in the video. Yang explained that disruption was not based on perception; rather, it meant that the allocated spectrum could not support the current video demand. Basically, it referred to packet loss beyond a certain threshold. Sen asked about the band used for edge detection, noting cases where noise reaches the level of the signal. Yang noted that when the transmission signal is comparable to noise, Jello combines energy detection and edge detection. If the transmission signal is so low it's indistinguishable from noise, it's considered free. Ratal Mahajan of Microsoft Research asked how close links were to one another. They used fixed-size guard bands to prevent interference in these experiments, but for heterogeneous devices in complex environments, adaptive guard bands could be useful.

PEER-TO-PEER

Summarized by Katrina LaCurts (katrina@csail.mit.edu)

- ***Contracts: Practical Contribution Incentives for P2P Live Streaming***
  Michael Piatek and Arvind Krishnamurthy, University of Washington; Arun Venkataramani, University of Massachusetts; Richard Yang, Yale University; David Zhang, PPLive; Alexander Jaffe, University of Washington

Piatek presented Contracts, a system that improves the performance and robustness of P2P live streaming. Current P2P live streaming systems (e.g., PPLive) offer no incentive for peers to contribute by sharing downloads. Piatek first discussed the possibility of using incentive mechanisms that rely on bilateral reciprocation (e.g., tit-for-tat), but noted that these are ineffective in live streaming systems, for three main reasons. First, peers are heterogeneous with respect to upload capacity, making it difficult to maximize capacity as well as limiting the number of users the system can serve. Second, there are limited trading opportunities between peers, as there are few "interesting" blocks in the system at a given time (due to the nature of live streaming), and peers far away from the source tend to receive blocks after everyone else. Third, there is no compelling reward in live streaming; clients can't download faster than the streaming rate.

Instead of relying on bilateral reciprocation, Contracts uses a global evaluation function to evaluate peers' contributions as well as the effectiveness of those contributions. These contributions are verified using cryptographic receipts. Peers who give more to the system are rewarded with robustness by being moved closer to the source. As a result, if the system becomes constrained, contributors fail last. Contracts also has defenses against collusion, which are presented in the paper. Piatek noted in the question-answer session that this is the main attack strategy against Contracts. Additionally, he agreed that if a large number of high capacity peers colluded and disconnected at once, there would be a significant service disruption, but connectivity could always be recovered by going to the source.

In the talk, Piatek presented results of his Contracts implementation in PPLive on Emulab. Results show that Contracts improves performance and strengthens contribution incentives over tit-for-tat. Further results in the paper examine Contracts' overhead, its performance vs. FlightPath (OSDI '08), the speed of its topology convergence, and its performance when over-provisioned.

- ***Experiences with CoralCDN: A Five-Year Operational View***
  Michael J. Freedman, Princeton University

Freedman presented a summary of his experiences from his five-year deployment of CoralCDN, a content distribution network (CDN) designed to make content widely available regardless of the publisher's resources. This talk focused on CoralCDN's interactions with the external environment (clients, origin domains, PlanetLab slivers, etc.) and usage patterns, rather than data from user feedback on their experience, which, as Freedman noted in the question-answer session, would be extremely difficult to collect. He also mentioned that it was a conscious decision not to provide active user support for the system, due to the amount of time that would consume. Currently CoralCDN takes very little time to maintain.

Freedman focused on three areas of CoralCDN: its naming scheme, its fault tolerance capabilities, and its resource management. CoralCDN's naming scheme provides a programmatic, open API for adopters that they began using as an elastic resource. However, it doesn't mesh well with domain-based access control policies, in particular those that use cookies (due to the nature of its naming scheme, CoralCDN's design can allow JavaScript-managed cookies from evil.com to interact with those from target.com). CoralCDN handles internal failures well, but has more trouble dealing with external failures due to problems such as misleading return values from origin sites. Finally, CoralCDN employs fair-sharing algorithms to enforce per-domain control over its bandwidth consumption (a reaction to some sites using CoralCDN—and thus PlanetLab—to serve a large amount of content), but suffers from a lack of control and visibility into the environment's resources. This motivates the need for lower layers to expose greater control over their resources.

Based on these experiences, Freedman discussed potential improvements to CoralCDN's design. Currently, CoralCDN is often used to resurrect old content or access unpopular content, neither of which it is particularly well suited for, as it wasn't designed for these purposes. It is also used to serve long-term popular content as well as to survive flash crowds

(Freedman noted that flash crowds tend to occur on the order of minutes rather than seconds, though it's hard to predict whether this will continue to be a trend). One could imagine redesigning CoralCDN to use little cooperation between nodes and focus on serving long-term popular content, or to focus on serving flash crowds and cooperate at a regional level, using more global cooperation only as insurance for origin failures. The latter could be done by leveraging the Coral hierarchy for lookup, and preliminary results for doing so are presented in the paper. In an attempt to reach Internet scale, Freedman announced the browser-based Firecoral, which can be downloaded at http://firecoral.net/.

- ***Whānau: A Sybil-proof Distributed Hash Table***
  Chris Lesniewski-Laas and M. Frans Kaashoek, MIT CSAIL

In this talk, Chris Lesniewski-Laas presented Whānau, a Sybil-resistant DHT that is oblivious to the number of Sybils in the network. Whānau involves two phases: setup and lookup. In the setup phase, nodes build routing tables using links from a social network. Relying on an assumption of sparse cuts between the honest nodes in the network and Sybils, random walks through the social network will return mostly honest nodes, and Whānau uses these nodes to build its routing table. The lookup phase is similar to lookup in a standard DHT. By keeping a sufficient number of fingers and keys at each node, with high probability Whānau can find a key in one hop. In the worst case, an honest node makes many connections to Sybil nodes, but Lesniewski-Laas noted in the question-answer session that Whānau allows for a certain fraction of these types of nodes, while still guaranteeing a one-hop DHT with high probability. He also noted that all nodes can utilize the same random walk length (this could be set as a system parameter); no node would benefit from choosing its own walk length.

Delving deeper into the construction of Whānau, Lesniewski-Laas presented the idea of layered identifiers. Layered IDs prevent the "cluster attack" on DHTs, where Sybils pick their IDs in such a way that they can overtake part of the DHT. Essentially, Whānau nodes choose multiple IDs—one per layer for log(n) layers—in such a way that there is a low probability of Sybil nodes clustering in any given layer. In the question-answer session Lesniewski-Laas mentioned that it would be harder to prevent this attack in multi-hop DHTs, as each hop moves a node further from its social network, and thus further from peers it trusts.

In the talk, Lesniewski-Laas presented simulation results verifying that popular social networks do exhibit the type of structure that Whānau requires (in particular, that they are fast-mixing), and that Whānau can deliver high availability even in a network with a large number of attack edges. Further results on layered identifiers, scalability, and churn are presented in the paper, along with an evaluation on PlanetLab.

WEB SERVICES 1

Summarized by Michael Chow (mcc59@cornell.edu)

- ***Crom: Faster Web Browsing Using Speculative Execution***
  James Mickens, Jeremy Elson, Jon Howell, and Jay Lorch, Microsoft Research

James Mickens described how in order to achieve a more responsive Internet we need to reduce user-perceived fetch latency. Prefetching has been suggested as a way to reduce latency. In Web 1.0 it was easier to prefetch data. There was a static graph connected by declarative links that had to be traversed. But in Web 2.0, the content graph is no longer static and the links are now dynamically linked. The new challenges to prefetching are handlers triggered by imperative JavaScript event handlers that cause side effects. This prevents the pre-execution of handlers. Thus, the prefetcher must understand JavaScript and it has to understand the side effects of execution fetches driven by user-generated inputs (e.g., clicking buttons or typing a search form). It is impossible to speculate on all possible user inputs. Prior solutions have used custom code.

Mickens then introduced Crom, a framework for creating speculative jobs. It is written in regular JavaScript with no modification to browsers. It is a generic speculation engine for JavaScript. Crom handles the lower-level details of speculative execution. It has to replicate browser state and rewrites event handlers. Replicating browser state requires cloning two pieces: the application heap and the DOM tree. To clone the application heap, it walks the object graph and deep-copies everything. To clone the DOM tree, the browser calls body.cloneNode() and then Crom has to do a fix-up traversal. Event handlers are rewritten to use the speculative contexts. It uses the JavaScript with(obj) function to insert obj in front of the scope chain. Speculative contexts are committed if two conditions are satisfied: the start state is equivalent to the application's current state and speculative input that mutated it must be equivalent to the real input.

The application must define two functions. It must define an equivalent function, which is a hash function over the global namespace, and a mutator function, which tells Crom how to change a new speculative context before running the speculative event handler. Mickens then demonstrated Crom speculating on a user Bing search. Crom has three speculation modes: full copy mode, checked lazy mode, and unchecked lazy mode. Full copy mode clones the entire heap for each speculation. It is always safe, but it may be slow. Checked lazy mode has lazy copy and parent tracking. It is also always safe, and parent mapping costs are amortizable across speculations. This may also be slow. Unchecked

lazy mode is fast. It is often safe in practice but, strictly speaking, it is unsafe without checked mode referencing.

The DHTMLGoodies Tab Manager was used to evaluate Crom. In evaluating Crom, they wanted to know if they could hide the overhead incurred from speculative execution in user think time. In unchecked lazy mode, pre-speculation costs sum to a trivial 24 ms. Committing the speculative context requires 77 ms. However, 99% of this is for committing the DOM pointer, which needs to be paid even for non-speculative execution, so this adds no overhead. In the checked lazy mode, there is a new source of overhead in creating the parent map pre-speculation, and at commit time it needs to fix state parent references. The pre-speculation time is 182 ms and the commit overhead was 5 ms, which is proportional to the number of objects cloned, not all JavaScript objects. Usually this is fine, but sometimes it is not acceptable for some applications. In conclusion, prefetching in Web 2.0 is different. JavaScript has side effects that need to be considered for speculative execution. Crom is a generic JavaScript speculation engine. Applications express their speculative intents, and Crom automates the low-level tasks for speculation. Crom can reduce user-perceived latency in loading dynamic pages.

Ryan Peterson from Cornell asked how Crom handles side effects from speculative executions on the server side. Mickens answered that the logs will show that these requests are speculative and that Crom relies on the developer to implicitly understand the side effects incurred on the server side (e.g., the developer should not speculatively execute purchases on Amazon). There is more on this in the paper. Wyatt Lloyd from Princeton asked about contention with running multiple JavaScript applications. Mickens replied that most JavaScript applications are not CPU bound, but it shouldn't be a problem when using unchecked lazy mode.

- ***WebProphet: Automating Performance Prediction for Web Services***
  Zhichun Li, Northwestern University; Ming Zhang, Microsoft Research; Zhaosheng Zhu, Data Domain Inc.; Yan Chen, Northwestern University; Albert Greenberg and Yi-Min Wang, Microsoft Research

Zhichun Li talked about the prevalence of Web services and how slow performance in these services leads to loss in revenue. An extra 100 ms delay at Amazon leads to 1% sales loss. Performance optimization is a hard problem. Web services are complicated, and knowing complete object dependencies is difficult: there is a lot of JavaScript, and services are hosted in data centers around the world. The metric used for performance optimization is page-load time (PLT). The user-perceived PLT is the page or portion with the most visual effects. PLT depends on a number of factors: client delay, net delay, and server delay. And each one of these delays comes from others such as DNS delay, TCP connection setup delay, and data transfer delay. Given the number of possible factors causing delay, there are a large number of possible performance optimization strategies, but there are also limitations to existing techniques. The A/B test, which uses controlled experiments, sets up a group of users to test the Web service. These tests are hard to fully automate, however, and are slow. Service provider-based techniques have problems with multiple data sources and object dependencies. Regression-based techniques usually require independence assumptions on delay factors of each object.

Li proposed a new tool for automated performance prediction with fast client-side prediction. It makes use of a timing perturbation based on dependency discovery and a dependancy-driven page load simulation. In order to extract dependencies, the authors used a lightweight black box–based approach that is browser-independent. Simple HTML parsing and DOM traversal are not enough, since there are event triggers, and object requests generated by JavaScript depend on corresponding .js files. The timing perturbation-based technique injects a delay through an HTTP proxy and sees how the delay propagates. HTML objects are special; they are stream objects, so they allow incremental rendering. The page is loaded slowly and the offset for each embedded object is measured. In order to simulate this process, the simulator loads the page according to a constructed dependancy graph and then adds corresponding delays, figuring out how long the new page load time is.

The WebProphet framework is a Web agent that extracts dependencies, predicts performance, and analyzes traces. The evaluation was on a Google and a Yahoo search. Checking the dependencies was done manually. In evaluating WebProphet, controlled experiments were run with a baseline of high latency experiments. The new scenario would use low latency connections, and a control gateway would inject and remove delays. In their PlanetLab setup, they used international nodes as the baseline and the new scenario for low latency nodes was nodes in the U.S. Looking at the results of the PlanetLab experiment, there was a maximal error of about 11%.

A possible usage scenario described in the talk was analyzing how to improve Yahoo! Maps. They only wanted to optimize a small number of objects. They evaluated 21,767 hypothetical scenarios in 20 seconds and found that moving five objects to the CDN results in a 14.8% speedup. Reducing the client delays of 14 objects to half the time results in a 26.6% speedup. Combining both leads to a 40.1% speedup or a reduction of load time from 4 seconds to 2.4 seconds.

Are there some services for which there are variable times depending on what kind of requests are made (e.g., a maps service)? If object and dependencies don't change, then the dynamic loading times will be the same. If not, they won't be the same. What about Internet routing delays? They are only looking at the client-side user perspective. Their tool only considers factors that the user sees.

- ***Mugshot: Deterministic Capture and Replay for JavaScript Applications***
  James Mickens, Jeremy Elson, and Jon Howell, Microsoft Research

In his second appearance as a presenter (and wearing a long, blond wig as part of his "intense commitment to sartorial flair"), James Mickens began by talking about when things go wrong with modern Web sites. Modern Web sites have event-driven functionality, with hard errors, such as unexpected exceptions, and soft errors, such as broken event handlers. When things go wrong, developers normally perform a core dump, stack trace, error log, etc., but it would be useful to them to see the path to how the error occurred. The developer cannot rely on users to report nondeterministic events either. The authors' solution is Mugshot, which logs nondeterministic JavaScript events, uploads an event log, and allows developers to replay the buggy program. Mugshot is a lightweight JavaScript library.

Mickens then presented an example of a page with a parent frame with a parent button and two child frames, each with a button. Each frame requires a copy of the logger.js file, but there is only one log maintained by the parent frame. In the example, the logger would need to log these two sources of nondeterminism: clicking the button and the return value of the date. Although it sounds simple, Mickens pointed out that for any DOM node/event name pair there can be, at most, one DOM 0 handler, while there can be an arbitrary number of DOM 2 handlers. Firefox calls the DOM 0 handler before DOM 2 handlers, which is difficult for Mugshot since Mugshot needs to run before any app-defined handlers in order to log these events. Multiple tricks, outlined in the paper, were used to get around both this and additional problems, such as IE having no capture phase, which makes logging GUI events tricky in IE.

In logging the value of loads, a mapping of a file name to bits may change from logging time to replay time. So a replay proxy is needed. At replay time, the developer uses the replay proxy as a Web proxy. On the developer machine, a transparent iframe is put on top of the page. Functions are interposed from the log, the log is fetched, and a VCR control is displayed. The log is stepped through on replay. In evaluating Mugshot, Mickens looked at how big the logs got. He showed that they grow at about 16 Kbps. The CPU overhead incurred from logging was only 2% on Firefox and about 7% on IE. This is important, since the user should not notice the logging.

Was there integration with the server-side, since applications run both in the browser and on the server? This was handled in many cases, because Mugshot logs HTTP headers, Ajax, and cookies. Would this work with other browsers such as Opera or Safari? Code used for Firefox could mostly be reused for Opera and Safari, since they are mostly DOM compliant. Was there really a need for the proxy and why couldn't they put it all in the client side? The proxy was needed because JavaScript code cannot inspect the cache, and there is no way for the Mugshot JavaScript to look in the cache after an image fetch.

WIRELESS 2

Summarized by Kevin Bauer (kevin.bauer@colorado.edu)

- ***AccuRate: Constellation Based Rate Estimation in Wireless Networks***
  Souvik Sen, Naveen Santhapuri, and Romit Roy Choudhury, Duke University; Srihari Nelakuditi, University of South Carolina

Souvik Sen presented a system that selects the optimal bit rate for a wireless channel between a client and the wireless access point. Choosing the optimal bit rate for a given wireless link is an important problem, since higher bit rates translate directly into better throughput for clients. However, achieving high throughput in a wireless link is not as simple as just picking the highest supported bit rate. If the wireless channel quality is poor, higher bit rates may result in excessive packet loss and reduced throughput for clients. Furthermore, unpredictable channel fluctuations caused by interference from other clients or environmental factors can affect a link's quality over time and space. This makes it difficult to predict the optimal bit rate for the link in advance.

Sen presented AccuRate, a system that uses physical layer information to improve bit rate selection, with the following intuition. First, the wireless physical layer encodes sequences of bits into a "symbol" in constellation space. AccuRate analyzes the dispersion between the transmitted and received symbol positions, where symbol dispersion is the distance between the received signals and the signal constellation. Small symbol dispersions indicate a strong channel that could potentially support a higher bit rate.

AccuRate arrives at the optimal bit rate for a given wireless link by measuring the symbol dispersions across many different bit rates. By replaying packets at increasing bit rates and measuring the resulting symbol dispersions, AccuRate determines the highest possible bit rate at which a packet could have been received successfully.

The system was implemented and evaluated using USRP hardware and GNURadio software with 802.11-like encoding and modulation schemes. In addition, the system is evaluated in simulation to characterize the performance with highly mobile clients. Sen explains that AccuRate is able to predict a packet's optimal bit rate 95% of the time when the packet is correctly received, and 93% of the time when only the preamble and postamble are correctly received. In terms of throughput, AccuRate is shown to achieve 87% of the optimal throughput.

Is it really necessary to estimate the optimal bit rate on every packet, or is it possible to perform more sporadic sampling? If the channel properties are known to be stable for a certain time period, it would be permissible to use the same bit rate for that time duration and then use Accu-

Rate again after this time elapses. Michael Freedman asked how frequently they need to run their algorithm, and Sen answered that you can use the rate for 100 ms and only calculate the rate once each time period.

- ***Scalable WiFi Media Delivery through Adaptive Broadcasts***
  Sayandeep Sen, Neel Kamal Madabhushi, and Suman Banerjee, University of Wisconsin—Madison

Sayandeep Sen presented an approach for achieving scalable delivery of high definition (5 Mbps or higher) media to clients at a wireless hotspot. Current approaches to media distribution over wireless networks rely on either unicast or broadcast. For unicast media delivery, the wireless access point receives one copy of the video and distributes *N* distinct copies of each video packet to each of the *N* clients. Unicast allows for individual packet retransmissions (via MAC layer acknowledgments) and physical layer rate adaptation, but is inefficient since each client needs its own copy of each packet. In contrast, wireless broadcast requires only a single copy of each packet for all clients. However, broadcasts have no channel feedback mechanisms (since MAC layer acknowledgments are disabled) and, consequently, offer no retransmissions or rate adaptation. To truly achieve scalable media delivery in wireless with improved performance, the authors sought to combine the best features of unicast and broadcast in their proposed system.

The key contribution of this work is the recognition that certain MAC layer functions such as retransmissions and rate adaptation can be improved with knowledge of the value of each packet to the application. Through value-aware adaptive broadcast, it is possible to retransmit valuable packets instead of packets with little or no value to the receiver. Sen presented MEdia Delivery USing Adaptive (pseudo)-broadcast, or Medusa, which offers rate adaptation, packet prioritization, and retransmission planning based on the priority of packets. A live demonstration of the Medusa system was shown to the audience, along with a comparison to wireless broadcast. The demo showed a significant increase in video quality.

Experimental results demonstrate that Medusa is able to achieve high PSNR (a measure of perceptual video quality) even as the number of clients watching the video increases from just one to over 20. This performance is comparable to unicast delivery to a single wireless client, which is considered to be near-optimal, and is significantly better than wireless broadcast.

An audience member observed that in the experimental evaluation, throughput decreases as more clients watched the video when multiple clients use unicast simultaneously for video delivery. Sen responded that because the link quality degrades with more clients, missed packets are not acknowledged. Michael Freedman said that this sounded like Scalable Reliable Multicast to him. Sen answered that they are working at the MAC layer, and the higher layers cannot do this.

- ***Maranello: Practical Partial Packet Recovery for 802.11***
  Bo Han and Aaron Schulman, University of Maryland; Francesco Gringoli, University of Brescia; Neil Spring and Bobby Bhattacharjee, University of Maryland; Lorenzo Nava, University of Brescia; Lusheng Ji, Seungjoon Lee, and Robert Miller, AT&T Labs—Research

Aaron Schulman presented a partial packet recovery protocol for 802.11 called Maranello. Partial packet recovery protocols try to repair corrupted packets, rather than retransmit complete packets, when an error is detected. This is done by retransmitting only the corrupted parts of the packet. Eliminating retransmission of unnecessary data can result in improved throughput and a better user experience. Maranello's goals are to provide partial packet recovery to 802.11 while maintaining compatibility with existing 802.11 deployments, be deployed incrementally, run on existing hardware, and require no extra bits for correct packets (since most packets will have no errors).

The Maranello system employs a block-based partial packet recovery technique. Packets are divided into 64-byte blocks, and checksums are computed over each block. If an error is detected, only the blocks containing errors need to be retransmitted. By retransmitting only the blocks that are corrupt, throughput increases because block repairs are faster than retransmissions, short block retransmissions have higher probability of delivery than larger whole-packet retransmissions, and early delivery avoids back-offs and low-rate retransmissions.

The block-based repair scheme is implemented in firmware so that it can be deployed incrementally on existing 802.11 cards. Relative to prior partial packet recovery designs, Maranello is the first to be implemented in commonly available firmware. Compatibility with existing 802.11 is achieved by having the receivers construct and send a negative acknowledgment (NACK) containing the block checksums before the transmitter decides to retransmit the entire packet. This ideally occurs after the short inter-frame space interval during which the transmitter expects an ACK. Thus, fast block-level checksums are required to meet this time constraint and the Fletcher-32 checksum is used for its speed.

Maranello was evaluated through trace-driven simulations and a real implementation using OpenFWWF and the b43 Linux driver. Simulation results indicate that successfully retransmitting earlier increases throughput. Also, an increase in throughput is confirmed, and packet recovery latency is reduced in a real deployment with other 802.11 hardware. Since most 802.11 implementations simply ignore NACKs, Maranello is able to fully interoperate with unmodified 802.11.

What is the size of the NACKs, since checksums are included in these packets? The NACKs can hold roughly 20 checksums, where each checksum is four bytes. Is it always possible to receive the NACK during the short ACK response window? As long as the data rate is greater than 12

Mbps, the NACK can be received during the normal ACK window. Are NACKs done during Clear Channel Assessment (CCA)? They are.

More information about Maranello, including firmware binaries (source code coming soon) can be found at http://www.cs.umd.edu/projects/maranello.

ROUTING

Summarized by Andrew D. Ferguson (adf@cs.brown.edu)

- ***Reverse traceroute***
  Ethan Katz-Bassett, University of Washington; Harsha V. Madhyastha, University of California, San Diego; Vijay Kumar Adhikari, University of Minnesota; Colin Scott, Justine Sherry, Peter van Wesep, Thomas Anderson, and Arvind Krishnamurthy, University of Washington

  Awarded Best Paper

Standard traceroute is one of the most commonly used tools for diagnosing network failures and anomalies. However, the asymmetry of paths on the Internet makes traceroute an imperfect instrument. For a datacenter operator attempting to diagnose problems seen by a client, a traceroute in the opposite direction may actually be necessary. Ethan Katz-Bassett presented the reverse traceroute project, which implements a novel technique for identifying the reverse path taken by a packet.

The reverse traceroute technique starts from the observation that routing on the Internet is based on destination. By maintaining an atlas of known Internet paths, the reverse traceroute tool only needs to stitch together the initial hops along the reverse path until it intersects one in the atlas. The initial hops in the reverse path are identified using the Record Route IP option, which is able to record the first nine routers encountered along a path. However, the host of interest is often further away than nine hops. To overcome this limitation, reverse traceroute uses other nodes as "vantage points," distributed on the PlanetLab testbed. By sending spoofed packets with the Record Route option from vantage points closer to the host of interest, it becomes possible to map the initial reverse hops.

Katz-Bassett presented an example where reverse traceroute was able to illuminate the cause of an inflated round-trip delay which had been noticed using normal traceroute. Correspondence with the network operators confirmed that reverse traceroute had identified a routing misconfiguration between Internap and TransitRail. Reverse traceroute can also be used to compute the latency along paths between arbitrary routers on the Internet. This ability was evaluated by comparing the calculated latencies with the published latencies of Sprint's inter-POP links. A Web interface to the reverse traceroute tool is available at http://revtr.cs.washington.edu.

Rodrigo Fonseca asked about packets with IP options being dropped when traversing the Internet and how that affects reverse traceroute. Katz-Bassett responded that support for IP options is improving and reverse traceroute uses alternate vantage points to try to get around networks which are dropping packets. Nick Feamster asked how they could take further advantage of the stability of paths in the Internet. Katz-Bassett noted that they are currently caching many parts of the path and are keeping an eye on other studies of the dynamics of paths in the Internet.

- ***Seamless BGP Migration with Router Grafting***
  Eric Keller and Jennifer Rexford, Princeton University; Jacobus van der Merwe, AT&T Labs—Research

Migrating a BGP router is currently a difficult but sometimes necessary process. The service it provides needs to be highly available and router changes generally must be coordinated with external organizations. Eric Keller identified the monolithic view of a router in which the hardware, software, and links are treated as one entity as the reason for this difficulty. By separating the components, Keller showed that it is possible to migrate the pieces independently without impacting the BGP session.

Router grafting requires migrating the four layers of a BGP session. The physical link can be moved by using a switchable, programmable network. The IP link is migrated by transferring the IP address, which, to routers in a BGP session, is simply an identifier. BGP relies upon long-running TCP sessions, requiring that the sequence numbers, queues, etc. be passed to the new router. Finally, the BGP state is migrated by dumping the existing configuration on the old router and importing it into the new router, while carefully tracking any updates that arrive during the migration. Keller presented a prototype implementation of router grafting for the Quagga software router and showed that the process is practical and transparent.

Keller was asked how this work relates to his group's previous work on migrating virtual routers (VROOM). He responded that this work is both complementary to and at a finer grain than the whole-router migration in VROOM. Nick Feamster asked if this technique can be applied to other routing protocols such as OSPF. Keller replied that migrating a link has a greater effect in OSPF installations because the shortest path may have changed and would need to be recomputed throughout the network.

DATACENTER NETWORKING

Summarized by Amin Tootoonchian (amin@cs.toronto.edu)

- ***ElasticTree: Saving Energy in Data Center Networks***
  Brandon Heller, Stanford University; Srini Seetharaman, Deutsche Telekom R&D Lab; Priya Mahadevan, Hewlett-Packard Labs; Yiannis Yiakoumis, Stanford University; Puneet Sharma and Sujata Banerjee, Hewlett-Packard Labs; Nick McKeown, Stanford University

Brandon Heller explained that ElasticTree aims to create an energy-proportional datacenter network from non-energy-

proportional components, because networking devices are not designed to be energy proportional. However, turning ports on/off affects their power consumption. ElasticTree compresses the network as much as possible based on the traffic by turning off unneeded ports/links and switches carefully in a scalable and resilient manner.

Input to the ElasticTree optimizer is the network topology, routing policy, power model, and the traffic matrix. The optimizer optimizes for power efficiency and later balances for fault tolerance and utilization. The solution is an active network subset and a set of flow routes. The rest of the talk was mainly about the three optimizers: formal optimization model (most optimal, least scalable), greedy bin-packer, and topo-aware heuristic (least optimal, most scalable). The optimizers come up with a single spanning tree, and the solution is modified to provide fault tolerance and a bound on utilization. ElasticTree was evaluated with simulations and using a running prototype with a Fat-Tree topology.

Costin Raiciu (University College London) raised the point that the graph shown for the traffic demand in a sample network was for the incoming traffic from outside, which does not necessarily reflect the inter-server communication pattern. Heller said they tried to capture that distribution by making some assumptions. Guohui Wang from Rice University asked about MapReduce workloads where all-to-all communications is necessary. Heller said their solution works regardless of workloads, but the underlying assumption is that the traffic demand does not need all the network capacity most of the time. Another attendee asked how ElasticTree would react to a sudden shift in traffic. Heller said that it depends on how fast switches/links can be turned on or put to sleep.

- ***SPAIN: COTS Data-Center Ethernet for Multipathing over Arbitrary Topologies***
  Jayaram Mudigonda and Praveen Yalagandula, HP Labs; Mohammad Al-Fares, University of California, San Diego; Jeffrey C. Mogul, HP Labs

Yalagandula started by arguing that today's datacenter networks require a low-cost flat network with a high bisection bandwidth. Ethernet, as has been pointed out by other recent works, is an appropriate choice: it is commodity, provides flat address space, and is self-configuring. However, having a spanning tree makes it not scale well. SPAIN uses commodity layer-2 switches and makes Ethernet scale under arbitrary topologies using multipathing via VLANs, and has end hosts spread load across VLANs.

SPAIN uses VLANs not for isolation, but for multipathing. Paths are computed and laid out as VLANs, and the end hosts choose a VLAN to send their traffic on. SPAIN wants to have as few paths as possible that provide enough reliability. The path computation module uses modified Dijkstra's shortest-path algorithm to find paths among edge switches with preference given to edge-disjoint paths (to provide higher bisection bandwidth). To reduce the number of VLANs, SPAIN merges the computed paths that do not form a loop. SPAIN was evaluated with simulations and a small testbed using Cisco datacenter, Fat-Tree, Hyper-X, and B-Cube topologies. SPAIN provides 1.6x–24x throughput improvement over STP in different topologies and, depending on the topology, may require a small or a large number of VLANs compared to the number of switches. Finally, the evaluation shows that incremental deployment of SPAIN is beneficial.

Amin Vahdat from UCSD asked about the assertion made in the talk about the deployability of VL2 and PortLand; he claimed that both are deployable today. Yalagandula argued that many data centers may end up buying cheap switches that do not provide features required by VL2. Also SPAIN works with arbitrary topologies (not only Clos or Fat-Tree). Albert Greenberg from Microsoft Research said that he had a paper about Monsoon (2008) which used VLAN tricks, but they moved away from Monsoon because of scalability and reliability: the L3 control plane provides built-in scalability and reliability which can be used. SRM said they are going to fix VLAN reliability issues using MSTP. Also, in the current solution, end hosts can work around the reliability concerns.

- ***Hedera: Dynamic Flow Scheduling for Data Center Networks***
  Mohammad Al-Fares and Sivasankar Radhakrishnan, University of California, San Diego; Barath Raghavan, Williams College; Nelson Huang and Amin Vahdat, University of California, San Diego

Mohammad Al-Fares said that datacenter workloads like MapReduce are often bottlenecked by the network. The standard approach is to use equal-cost multi-path routing (ECMP), but these routes are static and oblivious to link utilization. Hedera dynamically schedules flows under a dynamic traffic matrix in a network with any arbitrary topology. Hedera estimates demand and uses placement heuristics to find good flow-to-core mappings.

Hedera detects large flows, estimates their demand and pushes good paths for them into switches. For small flows, default ECMP load-balancing is efficient; therefore Hedera is complementary to ECMP. The placement heuristics considered are: global first-fit (where the first path which fits the flow requirements is chosen) and simulated annealing (probabilistic search for flow-to-core mappings). Hedera routes around failed components if a failure is detected, and in the case of the scheduler failure, default ECMP takes over. Evaluation on a simple testbed shows that simulated annealing outperforms global first-fit and is very close to the ideal case. Hedera also achieves close-to-ideal performance in a 120 GB all-to-all in-memory data shuffle. For larger topologies, simulations were used. They show that, again, simulated annealing is close to ideal, and Hedera is quite reactive to changes.

Someone raised the point that the power of two choices may significantly help to address the concern about ECMP. Al-Fares responded that static load-balancing is fundamentally flawed because it cannot satisfy dynamic traffic loads. Another attendee said large flows could be avoided in some scenarios, getting around the problems addressed in the paper. Al-Fares said the motivation was to provide a general solution without imposing any restriction on the traffic and applications. In a follow-up the questioner noted that because of small RTT in datacenter networks, the packets could be spread across multiple links and with the aid of a small reassembly buffer at the end-hosts, the problem could be alleviated. Al-Fares said they are currently working on that. Another attendee raised the point that if the number of flows is significantly larger than the number of paths, the distribution of flow sizes should not matter. Al-Fares said that they assume the traffic patterns obey the hose model. If the number of flows is far larger than the network can support, not much can be done. The point behind demand estimation is that the fair allocation can be found. David Anderson from CMU asked why the authors did not consider multicommodity flow routing with randomized rounding, since it is polynomial, fast, and reasonable. Al-Fares said that is part of their ongoing work.

IMPROVING MAPREDUCE

Summarized by Krishna Puttaswamy (krishnap@cs.ucsb.edu)

- ***Airavat: Security and Privacy for MapReduce***
  Indrajit Roy, Srinath T.V. Setty, Ann Kilzer, Vitaly Shmatikov, and Emmett Witchel, The University of Texas at Austin

Indrajit Roy presented Airavat. The MapReduce programming model is increasingly used today for large-scale computing. Many users upload their data to companies, which third parties later mine to provide better services to their users. For example, users might contribute their healthcare data, which may be mined to research new drugs or design cheaper insurance policies. However, a common fear is that the third parties that mine user data may target a specific user's personal data. Simply anonymizing the data is not sufficient because attackers can combine anonymized data with external information and de-anonymize them.

Airavat attempts to solve this problem, running untrusted code on the original data and yet ensuring that users' privacy is protected. The Airavat framework runs on the cloud, which includes modified MapReduce, DFS, JVM, and SELinux to provide privacy guarantees. Untrusted mapper programs from third parties are run in this Airavat framework, and yet Airavat guarantees bounded information leak about any individual's data after performing the MapReduce computation. Airavat prevents sensitive data leakage via network/storage by using mandatory access control, and Airavat controls data leakage from the output of the computation using differential privacy and a small collection of trusted reducers. Although Airavat supports only a small set of reducers (SUM, COUNT, THRESHOLD), it can still be used to perform many interesting computations. The paper used four benchmarks to demonstrate this. The overhead due to the modifications introduced by Airavat was less than 32% in these experiments.

Fang Yu (MSR) asked if Airavat was subjected to side-channel attacks. For example, the running time of the mappers can reveal some information about the computation—if something runs for five hours, it can be used to suggest something about the data. Roy responded saying that probabilistic channels and timing channels are exploitable and Airavat cannot prevent all such leaks. However, Airavat attempts to mitigate these problems by limiting access to system resource to the mappers. For instance, untrusted code cannot access the correct system time, and hence cannot directly know the time taken for running the mappers. Andrew Ferguson (Brown University) asked why the JVM was modified in Airavat. Why not instead reboot JVM after each MapReduce computation? This would easily prevent access to the heap after the job, etc. Roy agreed that this was a good idea, but suggested that this leads to higher overhead in scheduling the jobs.

- ***MapReduce Online***
  Tyson Condie, Neil Conway, Peter Alvaro, and Joseph M. Hellerstein, University of California, Berkeley; Khaled Elmeleegy and Russell Sears, Yahoo! Research

Tyson Condie presented MapReduce Online. MapReduce is tuned for massive parallelism and batch-oriented computations over massive data. But MapReduce is often used for analytics on streams of data that arrive continuously. There are two domains of interest in particular—online aggregation and stream processing. However, the MapReduce model is a poor fit for such computations, since MapReduce only produces the final answer and cannot work on infinite streams of data.

MapReduce Online is a new model based on MapReduce that can operate on infinite data and export early answers. The implementation proposed in this paper, called Hadoop Online Prototype (HOP), also preserves Hadoop APIs and implements pipelined fault tolerance. HOP pushes data from mappers to reducers concurrently with operator computation, which the reducers reduce periodically to produce intermediate results. These intermediate results are then published to HDFS. The paper presented extensive evaluation to show the benefits of generating such early results. For example, in an approximation query example, HOP was shown to produce good results, with low error, that was close to the final result. HOP code is available at http://code.google.com/p/hop/ and more information about the project is available at http://boom.cs.berkeley.edu.

John Dunagan (MSR) asked why not run a small job with sampled input, and what benefits HOP provides beyond sampling. Condie responded that this approach provides the output for one sampled point. If the programmer is not

happy with the results, then new jobs need to be run with different samples. However, in the case of HOP the programmers can watch the results as they are produced and decide when to stop the job. Y. Charlie Hu (Purdue) asked if HOP helps in resolving the problem of tuning and setting the right values to the myriad configuration parameters in Hadoop, so that an ideal set of parameter values can be assigned for executing a given job. Condie responded that HOP may not help in setting all the parameters, but it does help with setting the block size parameter in Hadoop.

WEB SERVICES 2

Summarized by Amin Tootoonchian (amin@cs.toronto.edu)

- ***The Architecture and Implementation of an Extensible Web Crawler***
  Jonathan M. Hsieh, Steven D. Gribble, and Henry M. Levy, University of Washington

Jonathan M. Hsieh presented a system named XCrawler. XCrawler is an extensible crawler service where consumers can set filters and only the documents matching the filters are delivered to the consumers. Hence, XCrawler develops an approach to outsource the resource-intensive crawling tasks of the consumers. The key insight behind XCrawler is to separate the act of crawling from application-specific tasks. This makes XCrawler a shared service that many consumers can use. This work focuses on providing flexibility for the consumers to specify the filter, making the crawler scalable to support many documents and filters, and achieving low latency to crawl even real-time data.

After presenting the core design, Hsieh also presented several optimizations that improved the scalability of XCrawler significantly. They evaluated XCrawler with three applications: a copyright violation detector, a malware detection application, and an online identity service.

Zhichun Li (Northwestern) asked if XCrawler crawled dynamic content, and if they considered using a complex language with more features for designing the filter language. Hsieh responded that XCrawler didn't support dynamic content or complex language as of now. But it would be possible to extend XCrawler to support these features. Stefan Saroiu (MSR) asked if they planned to extend XCrawler to supporting more general query types to enable future applications that may do more processing on the index rather than simply running regexes, and how one would refactor XCrawler to support such applications. Hsieh observed that their goal was to reduce the number of documents delivered to the consumer. The heavyweight processing of future applications can be done locally by the consumer on a small set of documents they receive on their own machines.

- ***Prophecy: Using History for High-Throughput Fault Tolerance***
  Siddhartha Sen, Wyatt Lloyd, and Michael J. Freedman, Princeton University

There has been significant work on BFT protocols in recent years, but the throughput of these systems is still low. In an effort to improve the throughput of Internet services that have a read-mostly workload, Siddhartha Sen presented Prophecy, which aims to extend the middleboxes that already exist in Internet services (such as load balancers) to act as trusted proxies. With this trusted proxy, Prophecy shows that it is possible to get a nearly fourfold improvement on the throughput of these systems. With a measurement study, the authors also show that many Internet services have mostly static workload (90% of requests are for static data in the Alexa top-25 Web sites) and hence can benefit from Prophecy.

The traditional BFT protocols provide consistency guarantees called linearizability, but provide low throughput. A way to improve the throughput for read workload is to store a collision-resistant hash (called a sketch) of the response on the servers, ask only one of the servers in the replica group to send the actual response, and ask the others to send only the sketch. The client can then compare the response with the sketches to decide whether the response is valid. This optimization means that only one of the replicas (not all) in the replica group needs to execute the request. A traditional BFT system with this optimization is called D-Prophecy. However, the problem with this approach is that the throughput can still go down when the session length that must be established between the client and the replica server is short. To resolve this issue, Prophecy introduces a sketcher (trusted proxy) that establishes the appropriate sessions with the replica servers, but the client only needs to maintain a long session with the sketcher. This leads to improved throughput. Sen also talked about how to scale Prophecy with multiple sketchers and presented evaluation results that showed Prophecy's performance for various amounts of reads in the workload.

Ryan Peterson (Cornell) asked why one should trust the proxy if one is not willing to trust the servers. Sen responded that many services already have semi-trusted middleboxes, and they can be easily extended with a small amount of code to act as the proxy. Peterson then pointed out that the proxy is a single point of failure, as the attackers can attack the proxy and start sending arbitrary responses. Sen agreed that this is a problem and if one is not willing to place this trust, then the D-Prophecy model is the most appropriate. Adding more proxies would also help, but then it only helps with fail-stop failures. Miguel Castro (MSR) asked whether not having an authenticated connection between the client and the proxy means that an attacker can inject arbitrary data. How does Prophecy deal with it? Depending on the application, Sen said, there should be some application-level security to deal with such issues. Some applications,

such as banking, need stronger mechanisms while other applications, such as Facebook, might be able to work with cookie-based authentication. Castro next asked why not use cookie-based authentication (and get rid of the sessions) between clients and the replica servers, and also eliminate the proxy? Sen pointed out that such a design would be a problem when there are larger replica groups, since the client will have to establish a session with each of the replicas.

MALWARE

Summarized by Adam Bender (bender@cs.umd.edu)

- ***Carousel: Scalable Logging for Intrusion Prevention Systems***
  Vinh The Lam, University of California, San Diego; Michael Mitzenmacher, Harvard University; George Varghese, University of California, San Diego

Millions of "interesting" network events, such as anomalies, routing changes, and packet drops, occur at a high rate. Network operators want to log such events, but any logging mechanism is often constrained, in terms both of memory and of bandwidth. The standard solutions for dealing with these limitations are sampling and calculating online summaries, which do not capture a complete event log. Terry Lam presented Carousel, a system for efficient, nearly complete event logging. While presented in terms of networking, specifically logging packet source addresses, Carousel has a wide variety of applications.

Lam presented the following model: packets from *N* sources are arriving at rate *B*. The router has a buffer that can store M bytes and write to a remote storage "sink" at a rate *L*. In many scenarios, *L* is much less than *B*, and *M* is much less than *N*. After describing why standard solutions are inefficient for complete event logging (they are a factor of $\log_n(N)$ worse, due to the coupon collector's problem), and why Bloom filters do not help, Lam also showed how an adversary can cause such collection never to terminate.

Lam then described the design of Carousel, which, given these constraints, eventually logs all events (given that the events are persistent) in nearly optimal time. Carousel stores packets in a buffer and employs a Bloom filter to prevent duplicate sources from being stored in the buffer. Carousel "colors" the packets and only logs packets of one color at a time. Thus only certain packets are added to the Bloom filter, preventing it from becoming full. After some time, the Carousel "rotates" to log packets of a different color and clears the Bloom filter. Lam presented an adaptive method of determining the right number of colors. Carousel has three components: partitioning the sources into colors (using the lower *k* bits of a hash function), iterating through the partitions every *M/L* seconds, and then monitoring the Bloom filters to adapt *k*.

The authors implemented Carousel in Snort and performed simulations of worm outbreaks; Carousel is nearly 10 times better than the naive solution of logging all unique sources. Lam also discussed an implementation in hardware. Because Carousel is probabilistic, it may miss some sources, and it relies on the assumption that events repeat, so that a missed event will eventually recur and be logged later. However, Carousel is within an expected factor of 2 of an optimal solution.

What did Lam mean by a "full" Bloom filter? A Bloom filter is full if its false positive rate is above some threshold. Zhichun Li asked how Carousel could be used to log distinct events from the same source, and Lam referred him to the paper for details on such experiments. Aditya Akella asked whether they considered other forms of scaling, such as using multiple snort boxes in parallel. Lam said they wanted solutions that would be efficient in both software and hardware.

- ***SplitScreen: Enabling Efficient, Distributed Malware Detection***
  Sang Kil Cha, Iulian Moraru, Jiyong Jang, John Truelove, David Brumley, and David G. Andersen, Carnegie Mellon University

Sang Kil Cha presented SplitScreen, a system for efficiently scanning a collection of files for malware. Traditional malware detection uses exact signature-matching on files. However, the number of signatures is growing exponentially (ClamAV had over 500,000 in 2009), requiring more memory and time to scan, as well as increasing the rate of cache misses. SplitScreen's insight is that it is very likely that only a small portion of the signature database is necessary to scan a set of files; this is suggested by a study of CMU email traffic that shows that less than 1% of all the signatures are needed to find all malware in the messages (in fact, 80% of malware can be caught using only five signatures).

To take advantage of this assumption, SplitScreen uses a feed-forward Bloom filter to reduce both the number of files that need to be tested and the number of malware signatures to test against. The feed-forward Bloom filter is initialized by taking a fixed-length (*n*-byte) segment from each signature and placing it into the all-patterns Bloom filter (APBF). SplitScreen takes a rolling hash of each *n*-byte segment of the target files and tests to see if any are in the APBF. If so, the file is placed into the reduced file set, and the matched bits of the APBF are copied into the matched-patterns Bloom filter (MPBF), which is the same size as the APBF. After scanning the files, SplitScreen iterates through all patterns to see which are present in the MPBF; these patterns compose the reduced signature set. SplitScreen then does exact matching on the reduced file set and reduced signature set. SplitScreen's Bloom filters are split into cache-resident and non-cache-resident parts and only access the latter when there is a hit in the former, making them more cache-efficient than classic Bloom filters. Cha presented results that show that SplitScreen has higher throughput (2x–4x) and fewer cache misses than ClamAV.

Another benefit of SplitScreen is that it can be run on smaller devices.

SplitScreen's reduced memory usage helps in this regard, but in this scenario propagating the large signature database is expensive. Cha showed how SplitScreen can be used in an on-demand way to allow low-power devices to perform malware checks: a server creates the APBF and sends it to clients; this is much smaller than the entire signature database. Clients then send the MPBF back to the server, which produces the reduced signature set and sends it to the client for pattern matching.

Seung Yi asked why the authors assume that only a small number of filters are necessary to match against all malware that may exist in a collection of files. Cha responded with evidence collected from two studies: one of a university email trace, in which less than 1% of signatures was needed to match all malware present, and a trace of 300 GB of malware, which required 20% of the signatures to find all samples. Wyatt Lloyd asked if SplitScreen was robust to a SplitScreen-aware adversary which uses old malware that is not in the SplitScreen database. Cha responded by saying that even if an attacker creates malware that matches many signatures, SplitScreen outperforms ClamAV. Yan Chen asked how SplitScreen transforms a signature into an entry in the Bloom filter, and David Andersen, a co-author, said that SplitScreen picks a fixed-length fragment of each signature. Stefan Savage asked how SplitScreen and other lazy-evaluation virus checkers can obtain new signatures when they do not have a network connection. Cha answered that SplitScreen is efficient even when running on a single host. Aditya Akella asked if the authors had thought of any other applications of feed-forward Bloom filters and cache-efficient designs. Cha said that he had not. The SplitScreen source is available at security.ece.cmu.edu.

- ***Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces***
  Roberto Perdisci, Georgia Institute of Technology and Damballa, Inc.; Wenke Lee and Nick Feamster, Georgia Institute of Technology

Roberto Perdisci presented work on observing and characterizing the network-level behavior of malware. While malware authors can evade antivirus by using obfuscation techniques, network properties are more difficult to hide: bots still need to contact a command and control (C&C) server, send spam, exfiltrate information, or download binaries. This suggests that a network-based approach can distinguish between legitimate traffic and malware traffic and thus identify compromised hosts in a network. The authors focused on HTTP-based malware, since it is growing in popularity and can bypass firewalls. The authors aimed to generate malware detection signatures. To reduce the number of signatures, they first classified malware into families based on network behavior. Network behavior was observed from HTTP traces of a large collection of malware samples. Then, the authors used a three-step clustering process to classify malware into families. The first step uses statistical features, such as the number of GET and POST requests and URL lengths, to form coarse-grained clusters. The second step employs structural characteristics to determine the difference between malware traces and uses this distance to split coarse-grained clusters. In the final step, the authors merge close clusters to recover from possible mistakes made in previous steps. The final step finds the centroid of each cluster using a token subsequences algorithm and merges clusters whose centroids are close. Once clustering is complete, the authors generate signatures from each cluster using a modification of the token subsequences algorithm.

To evaluate their techniques, the authors collected 25,000 samples of malware over six months. Perdisci presented results of an evaluation on a one-month subset that contained nearly 5,000 samples and from which the authors' technique produced 234 malware families with two or more samples (they discarded singletons). From these families, the authors generated 446 signatures over the course of eight hours. These signatures were able to detect a significant percentage of malware in separate traces collected in later months. The signatures also detected half of the malware that host-based antivirus products missed and had no false positives. Perdisci showed that the same signatures perform significantly better than those generated from a single-step clustering process or the latest host-based clustering technique.

Zhichun Li asked if Perdisci had examined types of traffic other than HTTP. Perdisci said this is part of his future work, but that HTTP was the most important type of traffic in enterprise networks. Had a malware author obfuscated network behavior to interfere with the presented technique? This was a very hard problem to solve; it might help to combine their technique with traditional anomaly detection techniques or thorough binary analysis. Yinglian Xie asked to what degree signature generation depended on clustering accuracy. Perdisci replied that correct clustering is critical. He also mentioned a signature pruning process that removes signatures that are likely to produce false positives. Areej Al-Bataineh asked what safety measures were used when running malware and if they have statistics on specific families of malware. Perdisci said they put an IPS in front of the infected host and did not focus on a specific family.

NETWORK PERFORMANCE

Summarized by Amin Tootoonchian (amin@cs.toronto.edu)

- ***Glasnost: Enabling End Users to Detect Traffic Differentiation***
  Marcel Dischinger and Massimiliano Marcon, MPI-SWS; Saikat Guha, MPI-SWS and Microsoft Research; Krishna P. Gummadi, MPI-SWS; Ratul Mahajan and Stefan Saroiu, Microsoft Research

Marcel Dischinger presented Glasnost, a tool that enables network users to detect traffic shaping. So far, half a million users from different countries and telecommunication regulators have used Glasnost. Designing Glasnost was challeng-

ing, because tests must be easy to use, short, and accurate. Glasnost performs active measurements in a controlled fashion; it compares the performance of two flows, the first flow emulating a realistic application (e.g., BitTorrent) traffic and the second one just varying the payload. Glasnost is able to detect port-based and content-based traffic shaping.

Glasnost runs on 20 servers on nine sites worldwide and is a part of Measurement Lab. Results collected from tests run by users show that 10% of BitTorrent tests indicated rate limiting over the 18 months of deployment. They also show that: major ISPs do rate-limit BitTorrent traffic. Rate-limiting is more common in the upstream direction and is based on both packet port and content; some ISPs only rate-limit some users or at peak hours. Finally, Glasnost enables automatic test construction using traffic traces.

Ming Zhang from Microsoft Research (Redmond) asked how they calculated the number of false positives and false negatives. They are using information from regulators (especially from Canada) as ground truth. How can the design be augmented to have it working in the case that ISPs detect Glasnost tests (i.e., treat Glasnost servers differently)? Centralized design is prone to whitelisting by ISPs, but having a decentralized design for Glasnost was challenging. The authors tried to make it hard for ISPs to detect tests by making the code available (so that they could run it on their own servers) and enabling users to make their own tests. Finally, Dischinger said that ISPs may choose not to take such actions, however, because it is bad for their reputation if they are caught.

- ***EndRE: An End-System Redundancy Elimination Service for Enterprises***
  Bhavish Aggarwal, Microsoft Research India; Aditya Akella and Ashok Anand, University of Wisconsin—Madison; Athula Balachandran, Carnegie Mellon University; Pushkar Chitnis, Microsoft Research India; Chitra Muthukrishnan, University of Wisconsin—Madison; Ramachandran Ramjee, Microsoft Research India; George Varghese, University of California, San Diego

Ram Ramjee argued that the middlebox approach to redundancy elimination does not work well in some scenarios: e.g., where end-to-end encryption is used, or in scenarios where the bottleneck is not the WAN (e.g., wireless links). EndRE takes an end-to-end approach to redundancy elimination to address these issues. Additionally, an end-to-end approach helps to save energy and can operate above TCP, which translates into latency gains.

EndRE's design goal is to opportunistically use the limited end-host resources: it is lightweight, fast, and adaptive; reduces memory overhead; and simplifies client decoding. EndRE proposes a fingerprinting technique called SAMPLEBYTE which is both fast/adaptive and robust. EndRE's evaluation shows that SAMPLEBYTE is an effective technique. Furthermore, analysis of 11 enterprise traces suggests that the median memory requirement of a client over 44 days is 60 MB, and bandwidth savings over two weeks was 26–34%, which, in turn, translates into notable energy savings. Finally, Ramjee concluded that EndRE is a promising alternative to WAN optimizers.

There were no questions.

- ***Cheap and Large CAMs for High Performance Data-Intensive Networked Systems***
  Ashok Anand, Chitra Muthukrishnan, Steven Kappes, and Aditya Akella, University of Wisconsin—Madison; Suman Nath, Microsoft Research

Ashok Anand argued that today's data-intensive networked systems require a cost-effective, cheap, and large hash table. He said using Flash SSDs is a good option because it is cheap and has a high read performance (10k random reads/sec), but is slow at writes. To deal with that they proposed a new data structure: BufferHash on flash which batches writes and performs sequential lookups (2x faster than random writes). Also, bloom filters are used to optimize lookups.

Their design has a two-level memory hierarchy, DRAM and Flash. To avoid unnecessarily going to flash during lookups, they use an in-memory bloom filter. To boost updates and insertions, they go to DRAM first and DRAM is flushed to flash eventually. Their evaluation suggests that BufferHash significantly outperforms traditional in-memory/on-disk hash tables, is best suited for FIFO eviction policy, and significantly improves the performance of WAN optimizers.

Jeff Mogul mentioned that phase change memory has different characteristics from Flash, and asked how much of their design would be useful with PCM. Anand replied that as long as there is asymmetry in read and write performance, their approach will work. Miguel Castro asked about a comparison to Berkeley DB that appeared in the paper, and how it was configured. Anand said it was configured the same way as CAM. Someone else commented that perhaps BDB needed to be rewritten for Flash.