REPORTS

Conference Reports

NSDI '14: 11th USENIX Symposium on Networked Systems Design and Implementation

April 2–4, 2014, Seattle, WA

Summarized by: Chen Chen, Michael Coughlin, Rik Farrow, Seyed K. Fayazbakhsh, Pan Hu, Chien-Chun Hung, Sangeetha Abdu Jyothi, Rishi Kapoor, He Liu, Feng Lu, Oliver Michel, Muhammad Shahbaz, Doug Woos, Qiao Zhang

Opening Remarks

Summarized by Rik Farrow

Ratul Mahajan, Microsoft Research, opened with a comment about how it was nice that we got to see Seattle when it wasn't raining. The rain held off until after the workshop concluded. Ratul went on to say that there was a new record in the number of submissions, 223, with 38 papers accepted for presentation over the next three days. The workshop included a new session track, operational systems track, on Thursday morning. The workshop this year drew over 250 attendees. Program Committee members had to review more than 27 papers each; fortunately, said Ratul, no one remembered (or at least commented on) the promise that each member would have fewer than 27 papers to review.

Ion Stoica, UC Berkeley, thanked Ratul for making his experience as co-chairperson a pleasant one. Ion announced that the award for Best Paper went to Mihai Dobrescu and Katerina Argyraki for "Software Dataplan Verification." The Community Award was given to EunYoung Jeong et al. for "mTCP: A Highly Scalable User-Level TCP Stack for Multicore Systems."

Finally, the Test of Time awards, for papers published at NSDI at least 10 years earlier that have had a lasting impact on their field, went to two papers (chosen by Tom Anderson, Stefan Savage, and Robert Morris of the Test of Time committee): "Operating Systems Support for Planetary-Scale Network Services" (better known as PlanetLab today), by Andy Bavier et al., and "Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks," by Philip Levis et al.

Datacenter Networks

Summarized by Sangeetha Abdu Jyothi (abdujyo2@illinois.edu)

Circuit Switching under the Radar with REACToR

He Liu, Feng Lu, Alex Forencich, Rishi Kapoor, Malveeka Tewari, Geoffrey M. Voelker, George Papen, Alex C. Snoeren, and George Porter, University of California, San Diego

He Liu presented REACToR, a hybrid Top-of-Rack (ToR) prototype that combines optical circuit switching and electrical packet switching to enable high-throughput networks. Although a 100 Gbps fat-tree would be an ideal choice for improving the throughput of an existing 10 Gbps datacenter network, this is an expensive option. The author mentioned previous work that attempted to improve throughput using optical circuit switching for large flows and drew attention to the fact that these techniques ignored the performance of short flows. In order to allow easy expansion of current 10 Gbps networks to 100 Gbps networks, the authors proposed the use of REACToR. REACToR combines the best of both worlds—it uses electrical switching, which allows buffering of packets for low bandwidth flows, and optical switching, which supports higher bandwidth for large flows.

In this model, the end hosts maintain a single queue for all packets belonging to low bandwidth flows destined for a packet-switched network and a per-destination queue for large flows to be sent on the circuit-switched network. REACToR sends Priority Flow Control (802.1Qbb) frames to pause and unpause flows at the end hosts. Since the link connecting user and REACToR is 100 Gbps, the optical bandwidth is limited to 90 Gbps in order to accommodate the sum of circuit-switched and packet-switched transmissions. Performance of the system was evaluated under various conditions. The TDMA scheduling is invisible to large TCP flows and guarantees fairness across flows irrespective of the schedule. The system also responds to demand changes in a fast and robust manner within 1.5 ms. The difference in performance between simulation and real world implementation was less than 1%. The system also performs well with a skewed workload (one flow constituting 50% of the traffic) and a very skewed workload (a single flow contributing 99% of traffic).

Changhoon Kim, the session chair, pointed out that benefits of the system depend on the speed and accuracy of traffic demand estimation. Liu responded that work on traffic estimation is ongoing. Currently, application-level information is used. Rik Farrow asked whether the experiment results were based on simulations only or involved real optical switches and the use of mirrors for the circuits. Liu replied that the results were based on a real implementation, and the Mordia switch does indeed use mirrors. Peter Hill asked about the diameter of the network. Liu responded that the testing was done on a small network with eight hosts. Liu also observed that scaling this system to accommodate thousands of hosts in a datacenter could be a challenge and requires further work.

Catch the Whole Lot in an Action: Rapid Precise Packet Loss Notification in Data Center

Peng Cheng, Fengyuan Ren, Ran Shu, and Chuang Lin, Tsinghua University

Peng Cheng presented cutting payload (CP), a mechanism that allows precise packet loss feedback. TCP faces several challenges—TCP incast, out-of-order packet arrival, etc.—and several mechanisms have been proposed to tackle each of these problems. But there exists no single solution that can mitigate all the problems associated with TCP. In order to improve the performance of TCP, three main challenges need to be addressed: (1) avoid TCP timeout caused by insufficient ACKs,

(2) distinguish packet loss and packet delays accurately, and (3) reduce packet loss detection time.

The authors address these challenges by cutting payload (CP). Switches use CP to cut off the payload of packets during congestion; only the headers are transmitted to the receivers. Upon receiving a payload-cut packet, the receiver parses the header and generates a PACK for the missing payload. The PACK is transmitted to the sender, which parses the message and triggers fast retransmission for the dropped payload. The authors showed that CP only increases resource usage by 2% and delays by 56 ns. The query completion time improved by 40% using this mechanism. Finally, Peng Cheng mentioned that CP is a TCP-complementary mechanism that is compatible with other versions of TCP used in datacenter networks.

Costin Raicui (University Politehnica Bucharest) focused on the importance of PACK and asked about the need for a response to the sender. Peng replied that PACK gave the sender more information regarding lost payloads. Costin then wanted to know more about the implementation of buffers in the switch. Peng responded that the switches had byte-oriented buffers of size 128 kB. The next questioner asked about a scenario where a bottleneck link is shared by a large flow that uses CP and several small flows which do not—how would the fairness criteria be met in this situation? Peng acknowledged that such scenarios were not tested. Changhoon Kim wanted to know more about the difference between ECN and CP and was referred to the paper. Another attendee asked about the choice to make changes at end-hosts and not use feedback from the switches. Peng answered that replies from switches would require network-wide changes whereas CP requires modifications at end-hosts only. Costin Raiciu returned to make another point: He referred to a scenario where the switch buffers contain only packet headers with no goodput in the network. Peng pointed out that CP has high goodput as demonstrated by the results in the paper.

High Throughput Data Center Topology Design

Ankit Singla, P. Brighten Godfrey, and Alexandra Kolla, University of Illinois at Urbana—Champaign

Ankit Singla put forward a systematic approach for high throughput datacenter design. The presentation focused on the design of throughput-optimal network topologies and dealt with two main questions: (1) How close can we get to optimal network capacity? and (2) How can we handle heterogeneity? In order to compare the performance of networks, throughput is computed as the solution to a linear program whose objective is to maximize the minimum flow in the network. Ankit pointed out that this is a more accurate measure of throughput than bisection bandwidth. Next, he presented an upper bound for throughput in a homogeneous network that is inversely proportional to the average path length. He showed that the throughput of random graphs is very close to this upper bound.

Next, Ankit dealt with the design of heterogeneous networks with multiple types of links and switches. The switches in the network are grouped into two clusters of high-degree switches and low-degree switches. The key challenges associated with the heterogeneous topology design for high throughput are (1) identification of the appropriate interconnection between the two category of switches and (2) determination of the best distribution of servers across these switches. Ankit showed that throughput improved initially when the number of cross-cluster links increases and then reached a plateau due to the upper bound imposed by the average path length. This allowed greater freedom in the cabling of heterogeneous networks. In addition, throughput was found to be optimal when servers were added in proportion to the port-count of the switches. Ankit pointed out that by using this technique of topology design, throughput performance of virtual layer 2 (VL2) topology could improve by 40%.

Tom Anderson (University of Washington) asked whether the workloads were realistic. Ankit responded that specific workloads cannot be representative for all datacenters. The traffic matrices used in the experiments were within two times the worst-case traffic matrix. As a follow up, Tom mentioned that optimizing the network for a particular traffic matrix is a dangerous trend and requested a clarification on the choice of the traffic matrix. Ankit replied that the workload is ideal for a generic high-capacity interconnect that can support a wide variety of applications. Hence, the results are applicable to any workload that is nearly uniform. Costin Raiciu inquired about the fraction of servers that contribute to the workload—the network is designed for 100% load, but the realistic loads could be 30%. Ankit replied that the network could be designed for the expected average load using the proposed mechanism, and the peaks could be tackled using hybrid approaches such as optical networks. Vyas Sekar (CMU) pointed out that the shortest path might not be optimal for traffic engineering. Ankit referred to results in the paper that show that the performance of packet level experiments using multipath-TCP is very close to the flow-level results obtained in simulations.

Debugging Complex Systems

Summarized by Rishi Kapoor (rkapoor@ucsd.edu)

Adtributor: Revenue Debugging in Advertising Systems

Ranjita Bhagwan, Rahul Kumar, Ramachandran Ramjee, George Varghese, Surjyakanta Mohapatra, Hemanth Manoharan, and Piyush Shah, Microsoft

Ramjee started by describing advertising (ad) systems as large distributed systems that are complex for two reasons: scale (millions of queries, users, publishers, ads) and the number of entities these systems interact with. The paper discusses revenue debugging: Why is revenue down at a given time anomalously, and how much does it cost?

Ramjee presented three interesting case studies/scenarios explaining the root cause for revenue anomaly. In the first scenario, a datacenter in Ireland had latency issues, which resulted in fewer ads being shown and consequent revenue decline. In the second scenario, revenue loss was caused by buckets using a new relevance algorithm. Finally, during the papal election, a lot of people were searching for "pope results," but there were very few advertisers who could show ads for these terms.

Ramjee said they focused on three aspects of these problems: a novel algorithm for root-cause analysis, attributes for derived measures (e.g., revenue per search), and they built a real-time time system for such root-cause diagnosis. For identifying the root cause of an anomaly, they picked a metric which deviated most from the expected value (JS-divergence). Throughout the talk, Ramjee gave examples of scenarios where derived measures such as cost per click (unlike individual metrics such as clicks or revenue) were able to detect anomalies by capturing correlations between the different metrics.

Ramjee also presented a demo of their tool: Adtributor. In their evaluation they show that their tool has an accuracy of more than 95% and saves troubleshooting time by 1+ hour per anomaly.

Evgeny Vinnick (Simon Fraser) asked whether they could expand this application to new markets and whether the code was available for experimentation. Ramjee replied that currently evaluation is done on four markets, but the system is running on more than 20 markets. Ramjee also said that the code is not currently open sourced, but they can consider that in the future. Someone asked whether the data collected by Adtributor could be used to create trending analysis: for example, could this tool be used to move an advertisement from x side to y side to earn more advertising revenue? Ramjee replied that there are lots of complementary problems, which are related but also distinct. They could use Adtributor but the question asked falls into a separate problem. The same person asked about network latency, and Ramjee replied that the network latency bubbles up to the top layer via the datacenter metric. Were there different datacenter metrics with a layer-wise approach? Ramjee replied to the smallest set question: If all your results are 100 elements long, but only one or two elements are the root cause, the smallest set is in regard to that. Ravi Bhoraskar asked whether they could apply this tool to any kind of troubleshooting: for example, could Azure use it? Ramjee replied that they could use it to detect failures and slowdowns in Azure or more generally. Ramjee mentioned the Bodik paper [Eurosys 2010] and added that these principles are fundamental, but he couldn't claim how easy it would be to port to other systems.

DECAF: Detecting and Characterizing Ad Fraud in Mobile Apps

Bin Liu, University of Southern California; Suman Nath, Microsoft Research; Ramesh Govindan, University of Southern California; Jie Liu, Microsoft Research

Bin Liu started by explaining how the ad ecosystem works. First, the app developer registers with an ad network, which then provides an ad plugin that the app developer incorporates into the app. The ad network selects an ad to be displayed to a user based on the user's location, interests, etc. When a user clicks on an ad, the ad network pays the app developer. Thus the app developer has an incentive to commit ad fraud by changing the way the ad is displayed (e.g., placement layout, invisible impressions) so that the user accidentally clicks on it. Bin Liu showed visual screenshots of such ad frauds. The focus of their work is to build an automated system that can detect such frauds.

The main challenge with building such a system is that it would need to scale to thousands of visually complex apps (an almost infinite number of pages and tens of clickable elements) and accurately and quickly identify the fraud. Other technical challenges include the sliding screen problem and the fact that the z-index is not available to identify hidden ads. Their automated system, DECAF, analyzed 50k phone apps and 1150 tablet apps. The system takes as an input an app, uses an automated UI navigation (Monkey), and outputs whether the app may contain placement frauds.

Someone asked whether they assume that DECAF can correctly extract all UI actions. Bin replied that they don't make that assumption, and there are indeed some apps that they can't recognize, mostly because the Windows Automation Framework can't identify them. Evgeny Vinnick asked whether they would be able to detect fraud if publishers use ads from different providers, like Google and Microsoft. Bin replied that they wouldn't be able to detect fraud for other providers. The system they designed needs a sub-checker for each provider. If another provider has a similar policy, they can use the same sub-checker but otherwise they will need to extend it. Someone asked whether the application developer could bypass their system, using techniques that Monkey can't detect such as a captcha (e.g., typing in three numbers to go to the next screen). Bin argued that they can design a new sub-checker to avoid application bypass. Microsoft is also working on adding a smarter ad-control checker. Bin added that their work is focused on speed, and they get better benefits than other tools. Ravi Bhoraskar asked why they couldn't use the page class to determine structure—for example, two different post pages on Reddit would be in the same class. Bin argued that the page class can't be used to determine things dynamically. It is necessary to compare pages at runtime.

I Know What Your Packet Did Last Hop: Using Packet Histories to Troubleshoot Networks

Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, David Mazières, and Nick McKeown, Stanford University

Nikhil Handigol started his talk by giving an example of a simple network bug—one that required hours of manual network debugging to identify the root cause and fix the issue. He argued that the current state-of-the-art tools in network debugging, tools like ping and traceroute, are tedious to use and require deep understanding of the tools and the system. Moreover, these methods are Band-Aid solutions—that is, they don't guarantee that they will be helpful in solving the outage. Nikhil emphasized that these tools provide close to zero visibility on what is happening in the system. He further argued that complete visibility of every event in the network is needed, which is challenging because this amounts to a huge amount of data to collect, process, and store.

Nikhil claimed that they can achieve complete visibility using packet histories (i.e., the path taken by a packet, modifications it encountered, and the current switch state) and a platform the authors built called NetSight. These histories can be used to diagnose faults in the system—for example, dropped/lost packets. Nikhil mentioned that naive implementation would result in huge overhead (of total bandwidth), and storing packet information would result in a huge storage costs. Nikhil described how they implemented such a solution using diff-based history and MapReduce-style parallel computation.

The authors developed four applications: NDB, netwatch (a live network debugger), nprof (a hierarchical network profiler), and netshark (a network-wide path-aware packet logger). Their evaluation demonstrated low overhead and the feasibility of complete network visibility. Nikhil mentioned that code is available at the following link: http://yuba.stanford.edu/netsight.

Aaron Gember (University of Wisconsin) asked how they ensure that they don't lose the tags on packets. Nikhil said that it is necessary to ensure that these tags are immutable. This should be enforced at the controller or proxy layer. Rodrigo Fonseca (Brown University) asked about the modifications needed at the switches to generate the post cards, and whether the method is very similar to sFlow, which would make it very slow. Nikhil replied that their work is different from sFlow in the way they correlate the exact state present on the switch. Unlike sFlow, the proxy adds a few extra actions to generate a copy of the packet header in the data plane and not in the control plane. Rodrigo followed up to ask whether the operations mentioned by Nikhil are standard OpenFlow actions. Nikhil mentioned that their prototype works with unmodified prototype OpenFlow switches; currently, OpenFlow doesn't have the ability to truncate the headers in hardware, making it highly inefficient as they copy entire packet headers. Rodrigo then asked whether the numbers mentioned in the talk, 31% and 7% compression, were with the prototype. Nikhil mentioned that these numbers are not for the prototype.

Timothy Wood (George Washington University) asked how they use these packet histories. Nikhil showed a few packet filter examples to detect reachability, isolation, and forwarding loops in networks. Timothy followed up by asking whether it is clear what these queries should look like. Nikhil mentioned that packet history appears like a string and that you apply a filter to this string using regular expression.

Libra: Divide and Conquer to Verify Forwarding Tables in Huge Networks

Hongyi Zeng, Stanford University; Shidong Zhang and Fei Ye, Google; Vimalkumar Jeyakumar, Stanford University; Mickey Ju and Junda Liu, Google; Nick McKeown, Stanford University; Amin Vahdat, Google and University of California, San Diego

Hongyi Zeng started his talk with a graph showing three common problems in Google datacenters. These problems are missing forwarding entry, forwarding loops, and black holes. Each of these trouble tickets is very expensive because it takes a very long time to debug them. Hongyi also mentioned that diagnosing these problems is difficult because it involves complex interactions between multiple protocols on the same switch and complex interactions between states on different switches. This arises from uncoordinated writes in the system. Hongyi mentioned that the static data plane verification does not work for datacenters for two reasons. First, in a large datacenter network the forwarding state is constantly changing, which makes it hard to take an accurate snapshot of the state for static analysis. Second, static analysis tools do not scale for large datacenters.

Hongyi said that they created a tool (Libra) that is fast and scalable and that can quickly detect loops, black holes, and other failures in large networks. First, Libra captures stable and consistent snapshots across large network deployments. Second, in contrast to prior tools that deal with arbitrarily structured forwarding tables, they substantially improve scalability by assuming packet forwarding based on longest prefix matching. The authors focused on the problem of obtaining a stable snapshot across thousands of switches. The gist of their solution is that they only consider the stable and consistent snapshots (discarding shady areas) and thus avoid false positives. Libra uses a divide and conquer algorithm that can be implemented using MapReduce to overcome the large scale of datacenters.

The data set is open sourced at http://eastzone.github.io/libra/.

Juan Francisco asked Hongyi to compare the previous paper's complete dynamic approach with their static analysis. Hongyi mentioned that these approaches solve two different sides of the problem. The static analysis is useful for checking functional problems that can be solved using a forwarding table. Performance problems are another class of problem that can be tackled only by using dynamic checking. Other sets of problems where incoming packets modify network state can only be solved in a dynamic state. Hongyi concluded that these approaches apply to different problems. Someone else asked about accounting for NTP inaccuracies, and whether there are any hardware switches that implement Lamport clocks (i.e., precisely the problem that solves the snapshot problem). Hongyi said that this problem can be solved if these systems have global clocks. NTP is more popular and deployed on the switches, and by adjusting the epsilon value they can achieve the same effect. Someone else asked whether they could gather events from the black box switches or required SDN switches. Hongyi answered that Google uses SDN-based switches. With traditional switches, it is hard to dump the state of the network, and you can't subscribe to the network. The final question was whether they could use the tool to do testing on controller software. Hongyi replied that it could be used as an independent checker to check the rules.

Verification and Testing

Summarized by Seyed K. Fayazbakhsh (seyed@cmu.edu)

Software Dataplane Verification

Mihai Dobrescu and Katerina Argyraki, École Polytechnique Fédérale de Lausanne

Awarded Best Paper!

Katerina Argyraki explained that software data planes are composed of packet processing elements. This is expected to make it easier to reprogram network functionalities. But in reality, bugs are present. The authors wanted to be able to take the binary along with the properties of interest, have a tool to verify the properties, and do so by adopting symbolic execution. Essentially, a section of code is represented using a tree that branches where the actual code branches. Exploring a path means examining the properties along the path. The problem with a naive application of symbolic execution is path explosion. The use of composition enables exploration of a subtree only once using domain-specific knowledge. The opportunity here is to utilize the pipeline structure of the data plane, which means usually there is no shared mutable state across data-plane elements for a packet.

The problem is that loops make reasoning about data-plane elements in isolation difficult (e.g., going over the IP options of a packet). The solution to this challenge is to share a small amount of state information between loop iterations. Another challenge is that there are many possible values that data structures may take. The solution here is to make explicit APIs by the programmer to enable data-access decomposition. A hash table and longest-prefix-match table are the proof-of-concept data structures that the authors implemented. The downside is that they cannot use dynamic memory allocation if they want to be able to verify the data structure. They have proved bounded instruction and crash freedom for Click data planes.

Costin Raiciu asked about the need to change the Click code and binary to make it work. Katerina answered that they did not need to change Click itself, but they had to change the loop structure part of metadata, which took a few lines of code. They also needed to extract data structures (e.g., for a NAT) using APIs. Costin then asked about whether the authors had any plans to build on this work, since what was done seemed practically limited. Katerina answered that they did support elements such as IP routing elements and NAT boxes. These are simple, but existing tools cannot capture them. Someone from Princeton asked what kind of interface would be needed to add to data structures. Katerina answered that key-value store interfaces were required to read or write or expire a value. Any well-defined interface would do. Someone asked about how easy it would be to determine verification requirements. Katerina said that the vision is that admins should not be doing this. Rather, the programmers should follow the authors' guidelines.

NetCheck: Network Diagnoses from Blackbox Traces

Yanyan Zhuang, Polytechnic Institute of New York University and University of British Columbia; Eleni Gessiou, Polytechnic Institute of New York University; Steven Portzer, University of Washington; Fraida Fund and Monzur Muhammad, Polytechnic Institute of New York University; Ivan Beschastnikh, University of British Columbia; Justin Cappos, Polytechnic Institute of New York University

Justin Cappos began by saying that applications like Skype are complicated to troubleshoot. Candidate solutions include using tools like ping or Wireshark, or trying to model apps and network elements to try to find bugs. These solutions are not completely practical. The insight in NetCheck is that humans make mistakes but not perfectly random mistakes. So we use tools like ktrace or strace for capturing system traces and then process and order these logs. This lets us diagnose problems. A trace is a locally ordered series of system calls. Each call has arguments and return values. Getting the exact global timestamps across traces captured in different locations is very difficult; we just need an approximation, though, as long as the extracted ordering is equivalent to the ground truth. NetCheck reconstructs what actually happened in the network using return values to infer the correct orders.

The network model is a simulated invocation of a system call. The runtime of the ordering algorithm is a linear function of the trace size. The fault classifier component of NetCheck decides what to output as the relevant information. We can configure what level of detail we want to receive. For evaluation of NetCheck, the authors reproduced reported bugs from bug trackers and found more than 95% of the bugs. NetCheck is extremely fast for logs larger than 1 GB.

Minlan Yu (University of Southern California) asked whether NetCheck needed the complete trace. Cappos answered no, NetCheck can deal with missing information. Wyatt Lloyd (Facebook) asked whether NetCheck works only for client-server applications and wondered about multi-party settings. Cappos answered that NetCheck could handle such scenarios. Someone from LinkedIn asked about performance issues, not just connectivity. Cappos answered that performance bugs were not targeted in this work. Someone asked how middleboxes come into the picture. Cappos responded that they can detect the problems in the middle as long as the applications are written to work with middleboxes.

Exalt: Empowering Researchers to Evaluate Large-Scale Storage Systems

Yang Wang, Manos Kapritsos, Lara Schmidt, Lorenzo Alvisi, and Mike Dahlin, The University of Texas at Austin

Yang Wang presented Exalt, a library that gives back to researchers the ability to test the scalability of today's large storage systems. Researchers usually do not have access to enough resources to test storage systems. The problem gets worse because sometimes the scale of the required test resources grows superlinearly with the system size. Quite often the I/O is the bottleneck, so we cannot simply add more resources. We need to abstract away data to scale the test method. The authors use data compression to solve this problem. The requirements for compression here include CPU efficiency, high compression ratio, and losslessness. Furthermore, the compression algorithm should be able to work with mixed data and metadata.

Our algorithm uses Tardis compression. Tardis allows data to be identified and efficiently compressed even at low-level storage layers that are not aware of the semantics and formatting used by higher levels of the system. This compression enables a high degree of node co-location, which makes it possible to run large-scale experiments on as few as a hundred machines. The authors used Exalt to identify several performance problems in HDFS and HBase.

Minlan Yu noted that the authors considered single nodes; what about the case of multiple nodes communicating with each other, and how efficient would the system be? Yang Wang answered that the work could not capture many nodes cooperating with each other. Wyatt Lloyd asked whether there were scalability restrictions with very large systems. Yang Wang answered that the authors could not guarantee full coverage, which would be the case with other tools, too. Someone from LinkedIn asked whether the authors considered a bottleneck caused by the hardware itself, such as network cards, when a lot of servers were emulated. Yang Wang answered that the implementation currently would not support that but that the authors were planning to consider a device-modeling approach to incorporate, for example, models of disks.

Security and Privacy

Summarized by He Liu (h8liu@cs.ucsd.edu)

ipShield: A Framework for Enforcing Context-Aware Privacy

Supriyo Chakraborty, Chenguang Shen, Kasturi Rangan Raghavan, Yasser Shoukry, Matt Millar, and Mani Srivastava, University of California, Los Angeles

Supriyo Chakraborty pointed out that some cell phone applications that provide utility to users read sensor data (e.g., to monitor user fitness); these applications can also infer sensitive information from the sensor readings (like user password) and hence violate user privacy. ipShield is an inference firewall that protects users from such attacks. Different from previous protection systems that statically restrict sensor data accessing, ipShield recommends privacy sensor accessing rules based on an inference whitelist/blacklist of higher-level privacy abstractions.

To use ipShield, a user first specifies an inference whitelist and a blacklist with priorities assigned to each inference. ipShield then will build the sensor-accessing rules based on an accuracy table. The table lists the inference accuracies that an application can achieve with different combinations of sensor readings. The recommended rules that ipShield outputs tend to maximize the accuracy of the whitelisted inferences and minimize the accuracy of the blacklisted inferences. Based on the recommendation, users can manually override the rules and create their own fine-grained policies.

Supriyo then talked about the complexity of implementing ipShield on Android systems, and showed the latency introduced and memory overhead of running ipShield. Source code of ipShield can be downloaded at http://tinyurl.com/ipshieldgit.

Jaeyeon Jung (Microsoft Research) asked how ipShield captures the correlations of different inferences, since information on one inference (such as location) might disclose information on another (such as activity). Supriyo said that in this paper, ipShield does not model it, but the team was in the process of integrating these correlations. Another person asked how ipShield tracks indirect data flow. For example, an application that has access to the GPS sensor can transfer the sensor data to another application via the storage card. Supriyo answered that as an extension to ipShield, they were trying to perform static analysis on the applications to track such data flows. Seungyeop Han (University of Washington) asked how ipShield could cover all possible inference types. Supriyo answered that crowd-sourcing could play a role here, but that is hard in general because important inference types in the future could be currently undefined.

Building Web Applications on Top of Encrypted Data Using Mylar

Raluca Ada Popa, MIT/CSAIL; Emily Stark, Meteor, Inc.; Steven Valdez, Jonas Helfer, Nickolai Zeldovich, and Hari Balakrishnan, MIT/CSAIL

Raluca Ada Popa explained that Web applications store sensitive data on servers, but it is challenging to protect the data from attackers who could have full server access. To handle this threat, Raluca presented Mylar, a framework that stores data encrypted on untrusted servers, where the data is only decrypted in a user's browser and presented via verified Web applications. On the untrusted server, Mylar stores user data with different encryption keys, yet the framework allows data sharing among multiple authorized users and data searching across multiple keys.

Raluca showed how Mylar works using a chat room example. First, Mylar generates the Web pages on the client side with certified Web application code (rather than on the server side). Since the server only acts as remote storage for signed code and encrypted data, it cannot tamper with the Web page. Second, Mylar manages shared data with a principal graph that is enforced by encryption key chains of certified keys. Users encrypt their data with different keys based on the principal graph. Third, Mylar introduces a new encryption scheme that enables multi-key search. If a user knows two keys, it can compute a "key delta" of the two keys and send it to the server, where the server can morph the encrypted data from one key to another so that searching is possible in encrypted form. With this setup, Mylar protects a user's data from other users and also fully compromised servers.

Implemented in 9000 lines of JavaScript and C++, the developer's effort to use Mylar is around 36 lines of code change on average, and the performance overhead introduced is also modest for Web applications. During the presentation, the PowerPoint slide failed to show the performance overhead figure, but Raluca managed to describe the results to the audience in her own words without the figure. The implementation can be downloaded from http://css.csail.mit.edu/mylar/.

Regarding the chat room example, one attendee asked whether two chat rooms created by the same user would share the same key. Raluca responded that two different chat rooms have different keys, and this depends on how the user/Web application chooses to handle it. Another attendee asked why some service providers that make profits from looking at user data would want to use Mylar. Raluca said that although there are clouds that make profits from user data, there are also clouds that rent resources to users. Mylar fits the second type; enabling the first type over encrypted user data is interesting future work. Finally, Sophia Xiao Wang (University of Washington) asked about Mylar's trust base. Raluca responded that those who use Mylar trust their own machines and Web browsers, not to mention the client-side code of the Web application developer.

PHY Covert Channels: Can You See the Idles?

Ki Suh Lee, Han Wang, and Hakim Weatherspoon, Cornell University

In this talk, Ki Suh Lee presented a covert timing channel called Chupja ("spies" in Korean) that works at the physical layer, with high bandwidth (100s Kbps), low bit error rate (less than 10%), and very hard to detect by upper-layer software. The threat comes from a passive adversary in the middle, equipped with commodity servers and NICs, trying to detect and monitor the communication between a sender and a receiver who want to exchange secrets. Both the sender and the receiver have full control of their own physical layers.

The design of Chupja is simple: It encodes information into a packet stream of the same packet length and inter-packet gaps (IPG) by varying the length of the gaps, where a slightly longer gap encodes a 1 bit and a slightly shorter one encodes a 0. Chupja is implemented as 50 lines of C code on top of SoNIC [NSDI '13], a software-defined network interface card that has full control of the physical layer.

Through evaluations, Lee showed that, throughput-wise, Chupja can achieve 81 Kbps of covert throughput over 1 Gbps overt throughput. Robustness-wise, the bit error rate increases with the number of hops on the route, but with a larger distance on the two coding points (the time difference between a 0 gap and 1 gap), Chupja can achieve a bit error rate of less than 10% even sharing the link with external traffic and, at the same time, still remain undetectable to software that runs on commodity servers that only have kernel timestamps for packet timing. The implementation can be downloaded from http://sonic.cs.cornell.edu.

Dongsu Han (KAIST) asked two questions: For the covert channel to remain undetected, the overt channel has to look innocent, but how is that done? Lee responded that the overt channel can simply transmit upper-layer application data as a cover. The second question was does it work with radar? Lee responded that Chupja's timing encoding scheme is general and can be applied to many communication channels.

cTPM: A Cloud TPM for Cross-Device Trusted Applications

Chen Chen, Carnegie Mellon University; Himanshu Raj, Stefan Saroiu, and Alec Wolman, Microsoft Research

Chen Chen stated that mobile devices have started to use trusted hardware, such as the Trusted Platform Module (TPM). However, protecting data with TPM across multiple devices is hard, because TPM-created keys are bound to a single TPM chip. cTPM (short for cloud-TPM) provides a solution for this. It embeds an additional root key pre-shared with the cloud. This enables seamless sharing of TPM-protected data with the cloud's assistance. It also provides additional benefits, such as a fast and large remote NVRAM storage and a trusted real-time clock.

One alternative to cTPM is to leverage secure execution mode (SEM), which is TPM's extensibility mechanism. However, SEM suffers from performance and engineering overhead, and lack of support on mobile devices. Instead, cTPM trusts the cloud, and provisions a unique random seed value pre-shared with the cloud. Based on this shared seed value, cTPM deterministically generates a "cloud root key" (CRK) and a "cloud communication key" (CCK). With these keys, the cloud can securely distribute shared keys to multiple cTPMs. The shared key is encrypted by the CRKs, and the communication channel is protected by the CCKs.

The cloud can also offer remote NVRAM storage. To handle disconnections and network latency, the cTPM also maintains a cache of the remote NVRAM storage. Each cache entry in the cTPM has a time-to-live field that dictates when the entry becomes stale. The untrusted OS and applications are responsible for re-syncing the cache; the synchronization protocol is protected with the CCK shared between cTPM and the cloud. Finally, a trusted clock can be implemented simply as a special NVRAM entry updated by the cloud. To offer this functionality, the cTPM proposes three new commands over the TPM 2.0 specification, and the protocol is verified with ProVerif.

Chen's implementation of cTPM is 12x faster than a typical hardware TPM for creating 2048-bit RSA keys; this is because software-based entropy source and crypto computation (on the cloud) are much faster than that of a TPM chip. Chen's team reimplemented Pasture [OSDI '12] and TrInc [NSDI '09] on top of cTPM as application demos.

One attendee mentioned that by storing the key on the cloud, the cTPM now provides a security property inherently different from TPM. A cloud compromise would inherently be a cTPM compromise. Chen responded that cTPM works as an addition on top of the current TPM design. While a cloud compromise would affect the root key pre-shared with the cloud (including the CRK and CCK), the unshared TPM-only root keys (e.g., the storage and endorsement keys) would remain safe. Also, making the cloud trusted is an active research area; emerging technology like Intel SGX could help with this issue.

Posters

First group summarized by Doug Woos (dwoos@cs.washington.edu)

Garbage Collection and Heap Growth Heuristics for Mobile Systems

Gabriel Arellano, Eduardo Dragone, Md. Jahid, Rogelio Ochoa, David Pruitt, Adrian Veliz, and Eric Freudenthal, University of Texas at El Paso

Garbage collection time represents a large problem for the performance and responsiveness of Android applications. Garbage collection is triggered in response to heap growth. The authors propose an alternative GC heuristic which allows the heap to grow and limits collection to at most once in a given time period (which they set to two seconds for evaluation). Their results were preliminary, but they found that with their policy, applications experienced no prolonged pauses and only modest additional heap usage.

GENI: Global Environment for Network Innovation

Vicraj Thomas, Niky Riga, and Sarah Edwards, BBN Technologies

GENI is a global-scale research testbed for networking and distributed systems projects. Similar to PlanetLab or VICCI, GENI allows researchers to provision virtual machines, OpenFlow-enabled switches, and WiMax base stations at many sites at US universities. Researchers can configure, via a simple drag-and-drop UI, multiple independent VLANs. GENI is now available for research use and is currently being used for multiple cloud-computing projects.

Online Censorship Resistance with freedom.js

Will Scott, Raymond Cheng, Arvind Krishnamurthy, and Thomas Anderson, University of Washington

The authors present freedom.js, a system for peer-to-peer communication in the browser. freedom.js is implemented in JavaScript and enables cross-platform communication. The authors have built several applications, including a file-sharing application, a VPN in which a user's traffic is routed through his or her friends, and activist.js, a system for avoiding censorship by enabling peer-to-peer access to blocked Web sites.

User Scripting on Android Using BladeDroid

Ravi Bhoraskar, University of Washington; Dominic Langenegger, ETH Zurich; Pingyang He and Michael D. Ernst, University of Washington

User scripting, as enabled by Greasemonkey and other site-modifying browser extensions, has become an important part of the way users interact with the Web. BladeDroid uses bytecode rewriting and dynamic class loading to bring the same functionality to Android applications, allowing users to modify application behavior without the assistance or permission of the application developer. The authors have written several extensions, including an ad blocker and an input automation system.

A Platform for At-Scale Wideband UHF MU-MIMO Systems

Ryan Guerra, Narendra Anand, and Edward W. Knightly, Rice University

The authors present a platform for research into wideband networking. Their card array consists of four Wideband UHF Radio daughter-Cards (WURCs), as well as four standard WiFi antennas; each radio is programmable using the WARP Software-Defined Radio system. To demonstrate the system, the authors implemented an application that measures and displays both UHF and WiFi channel capacity.

Enabling Multi-Layer Provisioning and Optimization for Core Transport Networks with Unified Packet-Optical Control Plane

Abhinava Sadasivarao, Infinera Corporation; Henrique Rodrigues, University of California, San Diego; Sharfuddin Syed, Chris Liou, and Sivaram Balakrishnan, Infinera Corporation; Andrew Lake, Eric Poyoul, Chin Guok, and Inder Monga, Energy Sciences Network; Tajana Rosing, University of California, San Diego

The authors present a system for provisioning and optimizing network paths in Tier-1 networks that include both packet-switching and optical components. The system is implemented on top of the Floodlight OpenFlow controller and allows the use of unmodified OpenFlow for data path programming. Rather than having to consider optical transport separately, the system maps practical transport concepts to OpenFlow flow abstractions. They have implemented the system on actual packet-based and optical routers, and demonstrate it with a bandwidth-policy-based optimization system.

Ib-KV: Using Infiniband Effectively for Key-Value Services

Anuj Kalia and David G. Andersen, Carnegie Mellon University; Michael Kaminsky, Intel Labs

The authors implement a key-value store on top of Infiniband's RDMA primitives, and demonstrate that by making two unconventional design choices they can achieve significantly better latency and throughput than similar systems (FaRM and Pilaf) on read-intensive workloads. First, Ib-KV client requests begin with a write operation rather than a read; clients write the request directly into server memory. Second, the server uses RDMA's built-in messaging system for its response, replying with a SEND message over an unreliable datagram transport. The primary benefit of these decisions is to avoid multiple round trips when making a request.

Second group of posters summarized by Chen Chen (chen.chen@inf.ethz.ch)

WiSense: A Client-Based Framework for Wireless Diagnosis

Ashish Patro, Prakhar Panwaria, and Suman Banerjee, University of Wisconsin—Madison

WiFi technology has been extensively deployed in homes and enterprises. Today, however, WiFi networks suffer from several performance issues, such as RF coverage, WiFi link/non-WiFi interference, traffic hotspots, channel contentions, etc. For users to better identify location-specific WiFi problems, the authors built an Android-based platform called WiSense, which can monitor WiFi network status without access points. Current WiSense implementation includes a NEXUS 7 device and an external USB card to collect fine-grained RF statistics. WiSense supports spectrum energy level monitoring, non-WiFi activity, per-channel airtime utilization, neighboring WiFi networks, aggregate per-link statistics, and active network measurements. The authors have demonstrated these features during demo sessions.

Pruning Masstree

Huanchen Zhang and David G. Andersen, Carnegie Mellon University; Michael Kaminsky, Intel Labs

Key-value stores are critical building blocks behind many cloud and network services like Facebook. The authors focus on building a space-efficient, high-performance key-value store that also enables range queries. The underlying data structure is called Masstree, which is a concatenation of layers of B+ trees that conceptually form a trie. The authors further designed pareto-optimal improvements to increase performance and reduce memory consumption, including designing more efficient memory allocation and garbage collection to save memory consumption, and serializing the B+ tree into a sorted array with binary indexing for higher performance. Preliminary results show that key-value stores could achieve 1.2 million items/sec with only around 200 MB memory consumed, and the designed improvements help increase the performance for range queries up to 3x.

Making the Live Network the Honeypot

Michael Coughlin, Oliver Michel, and Eric Keller, University of Colorado, Boulder; Adam J. Aviv, United States Naval Academy

Today's dedicated honeypots usually have different network topology, different applications, and different data than production networks. The differences not only leave the honeypot distinguishable from the production networks, but also make it difficult for network administrators to monitor the effects of attacks on production networks. In this work, the authors propose to combine live migration technique and the SDN to use the production networks as honeypots while isolating it from the attackers. According to their method, each time an attack is identified, the victim machine will be cloned to isolate the attacker from the production networks. The attack traffic will also be isolated from the production network by SDN. Finally, fake data are provisioned to replace actual data on the original machine to protect sensitive data. The authors have partially implemented the design based on KVM offline migration and SDN traffic dissection on top of Floodlight.

Towards an Open Middlebox Platform for Modular Function Composition

Shinae Woo, Korea Advanced Institute of Science and Technology (KAIST); Keon Jang, Microsoft Research; Dongsu Han and KyoungSoo Park, Korea Advanced Institute of Science and Technology (KAIST)

As the features of middleboxes continue to diversify, software-based middleboxes, which help consolidate multiple functionalities into a single box, have become increasingly popular over hardware-based middleboxes. However, the authors have identified three challenges for implementing software-based middleboxes. First, existing network stacks lack support for exacting flow-level information. Second, existing software stacks lack support for diverse transport-layer or application-layer events. Third, existing platforms do not provide programmable pipelines to process packets. In this project, the authors aim to provide a software-based middlebox development platform with three key building blocks—flow management, user-defined events, and module composition—to effectively address key issues in middlebox development.

Erwin: A Low-Latency Network Monitoring Platform

Jeff Rasley, Brown University; Brent Stephens, Rice University; Colin Dixon, Eric Rozner, Wes Felter, Kanak Agarwal, and John Carter, IBM Research—Austin; Rodrigo Fonseca, Brown University

In software-defined networks (SDN), the state-of-the-art network measurement system (sFlow) takes hundreds of milliseconds to collect a view of network conditions, which is much longer than installing a new route. The authors repurpose the port mirroring features provided by switches to efficiently produce traffic samples by oversubscribing the mirror ports. The mirror ports of different switches are connected to a collector, which is responsible for determining input/output ports for the packets, estimate the flow rates based on TCP sequence numbers, and answer queries about network status. The resulting measurement latency is demonstrated to be 250 μs–6.5 ms compared to current 100 ms–5 sec latency.

WiFi Mobility without Fast Handover

Andrei Croitoru, Costin Raiciu, and Dragos Niculescu, University Politehnica of Bucharest

WiFi networks are mostly static networks today, and mobile devices moving across different WiFi networks suffer from WiFi handover. The authors focus on reducing the handover duration for WiFi networks on mobile devices. The key idea is to leverage multi-path TCP to simultaneously connect to all available WiFi access points to avoid WiFi handover. The authors also demonstrate by experiments that the TCP congestion-control algorithm could well handle the interference caused by using a single channel and still provide high throughput. The authors also discuss ways to support multiple channels on mobile devices: using multiple NICs and using a channel switch to emulate multiple NICs.

Efficient Deployment of Network Management Policy Using Distributed Database Abstraction

Kamran Ali Akhtar, National University of Sciences and Technology, Pakistan; Muhammad Shahbaz, Georgia Institute of Technology; Saad Qaisar, National University of Sciences and Technology, Pakistan

The basic observation made by the authors is that the flow tables for controllers in software-defined networking (SDN) can be abstracted as relational databases. First, the controller would support Create, Read, Update, and Delete (CRUD) operations on the flow tables. Second, the flow tables on controllers must maintain Atomicity, Consistency, Isolation, and Durability

(ACID) properties. As a result, the authors propose to leverage existing Distributed Database Management System (DDBMS) schemes to manage flow tables for SDN and SQL as the interface to manipulate the flow rules. DDBMS could further provide features, including transaction validation, check-pointing, and deadlock mitigations, which are not present on current SDNs. The authors have used Mininet to build an implementation to validate the design based on POX and OpenFlow.

Toward a Precision Network Scripting with a User-Programmable Dataplane

Yohei Kuga and Takeshi Matsuya, Keio University; Hiroaki Hazeyama, NARA Institute of Science and Technology (NAIST); Kenjiro Cho, IIJ Research Laboratory; Rodney Van Meter and Osamu Nakamura, Keio University

Today, network processing is much more difficult than file processing on an OS for applications for network testing and diagnosis. Therefore, the authors built a network scripting framework called EtherPIPE, which provisions a character device interface to bridge the gap between user-mode UNIX command tools and the underlying network devices. Moreover, to provide highly precise timestamps for the packets sent and received, the authors further implement a NIC using NetFPGA to assign highly precise timestamps for packets for network diagnosis and to transmit packets at highly precise time points. The authors have demonstrated that the frameworks could support network programming with shell script tools and achieve time precision in microseconds.

Operational Systems Track

Summarized by Michael Coughlin (michael.coughlin@colorado.edu)

Network Virtualization in Multi-Tenant Datacenters

Teemu Koponen, Keith Amidon, Peter Balland, Martín Casado, Anupam Chanda, Bryan Fulton, Igor Ganichev, Jesse Gross, Natasha Gude, Paul Ingram, Ethan Jackson, Andrew Lambeth, Romain Lenglet, Shih-Hao Li, Amar Padmanabhan, Justin Pettit, Ben Pfaff, and Rajiv Ramanathan, VMware; Scott Shenker, International Computer Science Institute and the University of California, Berkeley; Alan Shieh, Jeremy Stribling, Pankaj Thakkar, Dan Wendlandt, Alexander Yip, and Ronghua Zhang, VMware

Teemu Koponen began his presentation by explaining that the presented paper represents five years of work by various researchers, especially those named as authors. He continued by discussing whether the issue of network virtualization is, in fact, already a solved problem by various technologies such as VLANs, MPLS, NATs, VRF, OpenFlow, etc. He argues that each of these technologies is only a point solution for a specific aspect of the network but does not address the network as a whole. Koponen then addressed datacenters directly, arguing that a network virtualization layer is hard to achieve, as tenant workloads are highly coupled to the infrastructure, and that a virtualization layer similar to what is provided to VMs by a hypervisor is needed for the network layer.

Koponen then introduced NVP (Network Virtualization Platform): The function of this network hypervisor system is to present a packet abstraction to VMs such that the network view seen by the VMs appears to be a physical network, which allows for applications to be run unmodified, and a control abstraction, which should allow for an ability to control the flow table pipeline on network hardware. NVP provides these abstractions, which allow for a tenant to create a logical data path, which, in turn, allows for the creation of any kind of logical topology. This system is implemented by using the virtual switches inside of standard hypervisors and connecting them using IP tunnels, with the switches being controlled from a central cluster.

Because this paper presents the product of five years of work, Koponen offered several lessons learned while operating NVP. First, some assumptions about logical networks cannot be made, because more complex workloads implicitly require more complex topologies that must be supported so that workloads are not modified; fortunately, this was addressed by the initial design decision to support arbitrary topologies. Second, OpenFlow proved to be insufficient for pushing state: Connections to OpenFlow controllers proved to be unreliable, which would lead to switches being left in an undefined state when a connection ended before completion. Third, OpenFlow is difficult to scale: Small operations still entail a large number of flow entries and may be redundant, and OpenFlow is highly coupled to switches. To address these last two issues, the authors are investigating a replacement for OpenFlow for the virtualization layer, but not necessarily the elimination of OpenFlow completely. Koponen concluded his presentation by restating the guiding principle of the project, which was to not modify workloads.

The first questioner asked whether the system obviates the virtual appliances that are built by networking companies and deployed in datacenters. Koponen explained that some appliances cannot be implemented in a distributed manner but can still be supported. He continued by stating that there will be pressure to implement these devices in a distributed manner, and he highlighted several applications built by VMware that illustrate this. Nodir Kodirov (University of British Columbia) asked if there was any experience with using the out-of-band network information debugging. Koponen replied that a large number of bits could be used in the encapsulation header that can and are used to debug workloads without affecting them. Marcin Kowalski (Amazon EC2) asked whether the researchers had looked at the performance implications of the software switches, in respect to 10G or 40G networks. Koponen replied that these implications had been investigated in the paper, including encapsulation optimization on x86 and Open vSwitch flow caching. Rick Schlichting (AT&T) asked whether there are any implications or lessons for virtualization in wide area networks. Koponen replied that wide area networks have not been investigated, but many deployments can span multiple datacenters, although the virtualization of wide area networks was not the goal of these deployments.

Operational Experiences with Disk Imaging in a Multi-Tenant Datacenters

Kevin Atkinson, Gary Wong, and Robert Ricci, University of Utah

Robert Ricci began by discussing disk images as one of the key tools for restoring a clean state to datacenter machines, and introduced his talk as a study of the disk imaging system of the Emulab datacenter, with an eye toward future research. He continued by explaining that the disk imaging system is an important consideration when provisioning a datacenter because of the need for large amounts of storage and bandwidth. He then asked three questions that he later answered in his presentation: What does the working set look like; what do the images themselves look like; and what are the key factors in preloading? The data set that was analyzed was collected over four years, consisting of 714 unique images from 1300 users and roughly 600 physical machines. Ricci noted that Emulab is not representative of all datacenters, but it provides an available data set.

Ricci then presented statistics gathered from the data set, starting with the number of requests for facility-provided images vs. user-defined images; this was a roughly even split, with 55% of requests being for facility images. Further analysis revealed that most users do not mix-and-match user and facility images and that heavy users tend to use user-defined images. Ricci continued by discussing the popularity of facility and user images. Analysis found that there was a small number of popular facility images, with the other facility images seldom used, whereas the user images were used more evenly by users, requiring a larger number of images to satisfy the same number of load requests. He then presented a scalability analysis based on the image usage trends: as the facility scales, all of the facility images will be used, because there is a limited number of facility images, but the number of user images will continue to grow.

Ricci described image content, noting that Emulab uses block-level similarity for users to create custom images, where users take a facility image and customize it, and then a diff can be taken from the base to store the image. Analysis of user images found that most were more than 60% similar to base images, which makes differential loading more useful. Ricci proceeded to discuss pre-loading of disk images onto disks. The first important consideration was the working set ratio (free pool of idle disks to number of images that can be pre-loaded), which allows for higher pre-loading effectiveness when this ratio is high. The second consideration was the rate of pre-loading an image; Ricci stated that pre-loading is more effective when investment is made in fast and scalable disk imaging. Ricci concluded by stating that a large number of images can be stored in slower storage, with a cache of popular images, and that facility images and user images are used differently and should be treated differently. All of the data and scripts used to write the paper are available at http://aptlab.net/p/tbres/nsdi14.

Vyas Sekar (CMU) asked whether Emulab has considered advertising images, in relation to how videos are advertised by sites like YouTube, in order to tweak popularity toward pre-loaded images. Ricci replied that Emulab provides a default image, but it is of poor quality and is not used by many users. However, this image is not aggressively advertised. Peter Hill (Microsoft Cloud) asked whether many images could be loaded onto servers to make use of extra space and reduce the number of active machines. Ricci replied that Emulab machines are pre-loaded with two images for these reasons, as many tenants will reside on the same disk. Katerina Argyraki (EPFL, and session chair) asked whether the designers would change the interface for user images if the system were to be re-designed. Ricci replied that a new design would keep the current interface, as other packaging options were made available, but these methods were not used by users.

VPN Gate: A Volunteer-Organized Public VPN Relay System with Blocking Resistance for Bypassing Government Censorship Firewalls

Daiyuu Nobori and Yasushi Shinjo, University of Tsukuba

Daiyuu Nobori began his presentation by introducing censorship firewalls, which are intended to censor access to the Internet, of which the Great Firewall of China (GFW) is a well-known example. A common method of circumventing these technologies is to use relay servers, but censorship authorities can easily block access to these relay servers, and resistance to blocking is difficult. VPN Gate's approach is to use thousands of volunteers and distribute the list to potential users using a central server. In order to prevent the authorities from blocking all volunteer addresses, innocent IP addresses (such as root DNS servers) are mixed into the list to force authorities to probe all addresses, and collaborative spying by volunteer hosts is used to identify authority IP addresses so that volunteer hosts can pretend to be innocent to requests from authorities.

To increase the number of volunteers in the system, Nobori explained, the relay program must be easy to use and install, and the program must function behind a NAT. These issues are addressed in the current implementation. The server program supports multiple VPN technologies, is based on the SoftEther VPN Server (http://www.softether.org), and supports NAT traversal. The system was launched on March 8, 2013, which led to a response from the authorities four days later using manual and then automated blacklisting, which was solved by the insertions of innocent IP addresses. The authorities then began to probe IP addresses, which led to the implementation of collaborative spying, where multiple abnormal connections to multiple servers are classified by a central server as an authority IP address.

Nobori presented an evaluation of the effectiveness of VPN Gate at evading censorship. At the start of the system's deployment, users were forced to attempt to connect to five different servers, on average, before a connection could be made, but after the deployment of collaborative spy detection on April 24, 2013, this decreased to 1.2 connections. Once the NSDI paper was completed, the GFW ceased attempts to censor the system.

Currently, the system is approaching 7,000 active servers and 400,000 active daily users, with 32,000 unique Chinese IP addresses, which is 13 times the estimated users of Tor. The system has also been deployed in Iran and North Korea, and the total bandwidth is approaching 3 Gbps. In conclusion, Nobori compared Tor and VPN Gate: They have a similar number of servers, but Tor has a much smaller number of unique Chinese users than VPN Gate, only supports certain technologies, and has weaker firewall resistance. Nobori concluded by saying that the system allows for people to help others overseas and may promote friendship, that no laws were violated in Japan, where the research took place, and that the system is based on open source software available at http://github.com/SoftEtherVPN/.

Aaron Gember (University of Wisconsin) asked how many different spy IP addresses from the authorities were observed. Nobori replied that approximately 2700 IP addresses were observed, but they were used for long periods of time and were reused. Matvey Arye (Princeton) asked how blocking of access to the central server list is prevented. Nobori replied that there are two mechanisms to access the list, via either an HTML table or an indirect protocol, which is accessible using VPN Gate proxy servers and HTTP mirror sites. Jon Howell (Microsoft Research) asked what would be the next move for the GFW. Could the GFW probe from many locations inside of the firewall? Nobori replied that it is difficult to predict what the GFW authority will do, but its ability to disturb the activity of Chinese users is limited. Someone asked whether it was possible to poison the server list to create a black hole, just as VPN Gate was able to poison the firewall address list. Nobori replied that this is possible, but the user can keep trying until a server is found. Someone asked whether the GFW performs deep packet inspection. Nobori said that the firewall uses three techniques: fake TCP RST packets, DNS IP reply poisoning, and IP address blacklisting. However, due to the high level of traffic, DPI is not scalable to all Chinese traffic.

Data Storage and Analytics

Summarized by Chien-Chun Hung (chienchun.hung@usc.edu)

Bolt: Data Management for Connected Homes

Trinabh Gupta, The University of Texas at Austin; Rayman Preet Singh, University of Waterloo; Amar Phanishayee, Jaeyeon Jung, and Ratul Mahajan, Microsoft Research

Trinabh Gupta started the presentation by talking about the need for data management of connected devices in the home environment. Some motivation scenarios include: (1) applications would generate data on a time basis and retrieve based on the time window, which means the data management system would need to support time-series, tagged data; (2) applications would access the data from multiple locations, and so the data management system would need to leverage the cloud servers for availability; (3) applications might share sensitive home data, and so the data management system would need to ensure confidentiality and integrity.

Gupta then presented Bolt, the data management system addressing the above-mentioned design concepts. There are four main features. First, end-users perform data encryption, while Bolt supports the append-only data abstraction "<timestamp, tag, value>" and batching data for efficiency. Second, Bolt supports data query based on the time-window "<start, end>," while the lookup and computation are performed locally at home. Third, Bolt leverages the cloud to support data availability. Finally, Bolt enables data security by decentralized access control.

The current Bolt implementation supports Windows Azure and Amazon S3. Its main performance gain over OpenTSDB is mainly due to data prefetching for subsequent data; the current data query is most likely to initiate the need for subsequent data records, batching for data query, and Bolt is 3–5x more space efficient than OpenTSDB.

Rik Farrow asked whether the system is mainly designed for Microsoft Home OS, and Gupta answered that Bolt can run outside of Home OS. George Porter (UCSD, session chair) asked how the authors collected the home sharing data from cameras. Gupta replied that currently all the results shown are based on synthetic data.

Blizzard: Fast, Cloud-Scale Block Storage for Cloud-Oblivious Applications

James Mickens, Edmund B. Nightingale, Jeremy Elson, and Darren Gehring, Microsoft Research; Bin Fan, Carnegie Mellon University; Asim Kadav and Vijay Chidambaram, University of Wisconsin–Madison; Osama Khan, Johns Hopkins University

James Mickens described the goal of this work as providing virtual block storage over commodity storage hardware in the cloud for cloud-oblivious applications using POSIX or WIN32 APIs, as if it is mounted locally, while achieving throughput greater than 1000 MB/s. Mickens outlined three key design challenges: (1) I/O dilation, that is, subsequent I/O operations generated by the end-host users may not be exactly subsequent when they arrive at the backend storage due to the delay dilation; (2) cross-rack I/O congestion limits execution parallelism; and (3) fsync() only returns when all writes complete.

This paper proposes Blizzard, which uses nested striping that assigns virtual disks (provided for the end-host users) randomly among physical disks. This random matching surprisingly achieves good performance. Blizzard also uses a Flat Datacenter Storage (FDS) style full bisection bandwidth network, and an RTS/CTS mechanism to avoid edge congestion. Finally, Blizzard delays later writes until earlier writes are flushed.

Keith Smith (NetApp) wondered if the paper's response to I/O dilation could be solved using prefetching data. Mickens pointed out that one of the key design goals for this work is to support unmodified applications, while prefetching does requires application modification to some extent despite the fact that prefetching does help alleviate I/O dilation. Someone from UBC said that fsync is evil, and that using dsync and osync should be preferred for ordering and durability. Mickens reminded the questioner that a design goal was not to modify existing applications. Someone who works on DynamoDB (Amazon) wondered about reads when writes were still buffered. Mickens responded that they deal with that issue. Ashton Goel (University of Toronto) worried about applications, such as mailtools, that expect durability. Mickens answered that some applications are fine with delayed durability and are willing to engage in that tradeoff.

Aggregation and Degradation in JetStream: Streaming Analytics in the Wide Area

Ariel Rabkin, Matvey Arye, Siddhartha Sen, Vivek S. Pai, and Michael J. Freedman, Princeton University

Ariel Rabkin began by showing that backhaul bandwidth is intrinsically inefficient, because it is sometimes under-provisioned and sometimes over-provisioned for streaming data analytics. The authors addressed this problem and optimize the use of WAN bandwidth by proposing a data streaming framework, JetStream, with aggregation and degradation mechanisms. Aggregation is done by merging records with the same property, whereas degradation is achieved through sampling and filtering data according to the current bandwidth capacity.

To realize its goals, JetStream requires a few characteristics for data abstraction: that data be updateable (locally and incrementally), reducible in size (with predictable accuracy cost), and mergeable (without accuracy penalty). A key design component is the data cube, which is the multi-dimensional array indexed by a set of dimensions. Cubes can be rolled up and unify the storage and data aggregation. Another key design for supporting auto degradation is to coordinate between network operators so that the degradation can be achieved to match the bandwidth capacity appropriately, although there is a tradeoff between bandwidth saving and accuracy.

Someone from Microsoft Research asked how to quantify the accuracy penalty during degradation. Rabkin confirmed that the accuracy penalty is generally well-behaved based on the granularity of sampled data records, e.g., from second level to minute level. The same person followed up with some scenarios—e.g., calculating the maximum values within a data set, the accuracy penalty is therefore not clear during degradation. Rabkin replied that in that kind of case, degradation could be achieved by dropping the low-ranked values, instead of coarsening the data granularity. Kurt Colovson (VMware) suggested that they might want to impose some hysteresis. Rabkin replied that they could add something for stability in the controllers.

GRASS: Trimming Stragglers in Approximation Analytics

Ganesh Ananthanarayanan, University of California, Berkeley; Michael Chien-Chun Hung, University of Southern California; Xiaoqi Ren, California Institute of Technology; Ion Stoica, University of California, Berkeley; Adam Wierman, California Institute of Technology; Minlan Yu, University of Southern California

Ganesh first pointed out that approximation analytics, which trades complete results for quicker responses, is becoming more trendy and general in big data analytics. What makes scheduling for approximation jobs challenging are the certain requirements (e.g., deadline, accuracy) and the existence of stragglers. Specifically, should the stragglers be mitigated by speculation, dynamically prioritizing between original and speculative tasks, while meeting the deadline/accuracy requirement?

Two heuristics for straggler mitigation are Greedy Speculation (GS, which is more aggressive) and Resource Aware Speculation (RAS, which is more conservative). Ganesh talked about the guidance from analysis model, which shows that optimal scheduling strategy is to start speculation conservatively in the early stage of a job, then turns aggressive as the job gets close to completion. Based on the guidance, the authors propose GRASS, which starts with RAS first and switches to GS as the job is about to complete, while the switching point is learned from job samples. Experimental results obtained from Hadoop and Spark implementations suggest GRASS improves deadline and accuracy by 47% and 38%, respectively.

One person asked about the task duration's dependency on data content. Ganesh replied that GRASS assumes each data unit contributes equally to the computation volume within a job. Moreover, GRASS results show that when the estimation accuracy is low, sticking to RAS is better than switching to GS.

Interpreting Signals

Summarized by Pan Hu (panhu@cs.umass.edu)

Bringing Gesture Recognition to All Devices

Bryce Kellogg, Vamsi Talla, and Shyamnath Gollakota, University of Washington

Bryce Kellogg started his presentation by asking for new interaction technology that goes beyond mouse and keyboard. He focused on gesture recognition in this presentation. After pointing out the drawbacks of Kinect-based gesture, such as failing to work in non-line-of-sight scenarios and consuming too much power, he demonstrated in a mobile application the authors' ultra low-power gesture recognition system using hand gestures near his pocket, where he kept his mobile device, to change songs and increase and decrease volume.

Bryce further explained the technology detailed behind the demo. He introduced AllSee, the first gesture recognition system that runs without batteries, their prototype leveraging the ambient signal from RFID and TV signals with the intent of expanding into WiFi and cellular. The hardware architecture consists of a wireless signal receiver, some digital logic, and an energy harvester circuit. Bryce pointed out that traditional receivers that actively generate carrier waves consume too much power. Since the power AllSee gets is from ambient, the energy harvester is very small (~40uW) and it is not possible to employ traditional radio on these kinds of devices. The power budget also limited the capability of the AllSee receiver. By using a technology that is similar to Ambient Backscatter introduced in SIGCOMM '13, Bryce managed to get the amplitude information from the receiver. This poses another challenge to the digital signal process since it contains neither frequency nor phase information, and thus traditional gesture recognition technology based on Doppler or angle of arrival could not be implemented on AllSee. What's more, Bryce pointed out that it is unlikely that a machine-learning algorithm would be implemented on battery-free devices.

Their solution was to build an amplitude library of gestures and focus on the trends. Then they proved their algorithm was simple but accurate—94% gestures were correctly classified. By duty cycling the microcontroller, Bryce pushed the energy consumption of AllSee to merely 30uW, which fit into the power budget perfectly.

Multiple people showed deep interest in their technology. Deepak Ganesan (University of Massachusetts) first asked about the working range and orientation sensitivity of AllSee. Bryce replied that the working range is about 2.5 feet. Direction could have an effect on the performance of AllSee since it uses a dipole antenna, but the effect could be cancelled through calibration. Deepak then asked whether it was always possible to harvest 40uW from the environment—for example, in rural areas. Bryce replied that although TV signals are quite pervasive, they are looking to extend the technology to cellular and WiFi so that energy harvesting is guaranteed. Pengyu Zhang (University of Massachusetts) asked whether the sampling rate (200Hz) would limit the accuracy of gesture recognition or not. Bryce answered that 200Hz works well for most scenarios. It also provides a good balance between power consumption and accuracy. Joshua Smith (University of Washington) asked about future work. Bryce told us they hope to integrate AllSee into commercial off-the-shelf devices.

3D Tracking via Body Radio Reflections

Fadel Adib, Zach Kabelac, Dina Katabi, and Robert C. Miller, Massachusetts Institute of Technology

Fadel Adib introduced a motion tracking system that uses reflected signals only. To answer the question "Can we see through walls with wireless signals?" he introduced the WiTrack, a system that tracks 3D motion of a user behind a wall by using radio signal reflected back from the user. After that, he demonstrated his system with a video of a man walking on a white line. A laptop in another room showed the real time position of the man, which was very accurate. He also showed that WiTrack could track body part movements by gesturing with an arm to turn off a light or a TV. At the end of the video he shows that the man could turn off the light in another room, which is impressive and attendees started to laugh. Fadel concluded that WiTrack could achieve centimeter-level accuracy, which is suitable for gaming, gesture control, rescue, or monitoring the elderly.

Fadel introduced the architecture of WiTrack after the interesting demo. The first step is to measure the distance in order to get the position of the user. This can be done by computing the time of flight of the reflected signal, but it requires a very high sampling rate. In addition, this method is susceptible to noise, which is not suitable for sensing behind a wall. Instead, Fadel introduced frequency modulation continuous wave radar, which sends out a chirp signal rather than a signal of a single frequency. By analyzing the frequency difference between the sent and reflected signal, WiTrack can get very accurate distance measurements. Fadel also shared one of the challenges in implementing the system: multipath. Not only does the human reflect the signal, other objects including the wall also reflect. However, background signals can be eliminated by doing subtraction by assuming that the user is moving while the environment is not. However, it is still possible to see multiple peaks due to dynamic multipath such as people interacting with a table. This can be solved by always looking at the shortest path because the direct path is always shorter than reflected paths.

After obtaining the distance, Fadel explained how to achieve localization by putting three antennas for trilateration. The WiTrack system works from a bandwidth of 5.5 to 7.2GHz with a transmit power of 0.75mW. A motion tracking system is used to serve as ground truth. Experiment results showed that WiTrack could achieve 10 cm, 13 cm and 21 cm in three directions. Fadel also showed that WiTrack could have achieved accurate motion detection with an orientation error of 11 degrees. The accuracy of fall detection is also very high.

Joshua Smith (University of Washington) asked about the regulation of this kind of device since it may involve privacy issues. Fadel answered that people could try to block these signals or the government could regulate their usage. Joshua also asked whether it is easy to filter out background signals. The answer is not easy because there are other moving objects such as fans in the room. Fadel also pointed out that steering the RF signal beam might help to filter out the background signal. Deepak Ganesan asked whether segmentation is needed in the gesture recognition and Fadel answered yes. Jie Liu (Microsoft Research) asked about the difference between WiTrack and traditional radar and sonar. Fadel pointed out that traditional radar fails to deal with multipath in indoor scenarios; what's more, it cannot provide centimeter-level accuracy. Jie Liu also asked about the possibility of trying to distinguish between people. Fadel replied that it could be done by looking at the reflected pattern. Finally, Jie asked how the properties of the wall might affect the wireless signal and its accuracy. Fadel answered that the wall may have a small impact on the accuracy although they haven't tested on different kinds of walls yet.

Epsilon: A Visible Light Based Positioning System

Liqun Li, Microsoft Research, Beijing; Pan Hu, University of Massachusetts Amherst; Chunyi Peng, The Ohio State University; Guobin Shen and Feng Zhao, Microsoft Research, Beijing

Localization is a hot (and old) topic throughout the last decade but Guobin Shen presented something new. Recent localization technology based on WiFi, cellular, FM, or magnetic fields either failed to provide high accuracy or required a complicated infrastructure. Guobin presented Epsilon, an indoor localization system based on visible light. Guobin suggested that LED lighting is the future light for industry due to its high efficiency and longer life. And LED as a semiconductor component intrinsically supports rapid switching on-and-off. By taking advantage of this character, current dimmer switches can change the light intensity smoothly by switching on-and-off using different duty cycles. Inspired by this fact, Guobin said we could use LEDs as beacons. The advantage of using visible light includes higher beacon density due to the pervasive existence of LED lights and getting humans into the loop. The system leverages the current infrastructure and thus has minimal cost.

Guobin then explained how to implement this system. Epsilon adopted a model-based rather than a fingerprint method to reduce the cost needed to deploy the system. Since both the LED and light sensor are directional to some extent, Epsilon models the irradiation angle from the LED and the incidence angle to the light sensor. Evaluation shows that the accuracy of the model is pretty good when both angles are small but that relative error increases as the angles increase. Guobin also shared some practical design considerations for Epsilon. Firstly, commercial LEDs could not be modulated with a frequency higher than 100 kHz, and they need to modulate faster than 200 Hz to avoid perceptible flicker. What's more, power line frequency could also affect the positioning system. By carefully choosing the frequency and modulation method, Epsilon managed to meet all these needs. Guobin also talked about how to deal with a scenario with only one LED light. Epsilon could involve human motions to improve the position results. For example, by letting users tilt their cell phones while using the IMU, it would still be possible to get the position. Experimental results showed that the 90th percentile accuracy is 40 cm, 70 cm, and 80 cm in three dimensions.

Joshua Smith asked about privacy issues of this system when compared with other ones. Guobin replied that Epsilon might have better privacy preservation because light can be more easily blocked than wireless signals. Joshua also asked whether it is possible to achieve higher accuracy by increasing the modulation frequency. The answer was no, because Epsilon only uses signal strength to measure distance in the model, which is irrelevant to frequency. Deepak Ganesan asked whether it is possible to achieve higher accuracy by deploying more LED lights. Guobin answered yes, although Epsilon will always choose the four LED lights with the highest signal strength. Deepak also asked whether the orientation of the user would affect the accuracy or not. The answer was yes, but they have taken this into consideration when evaluating their system.

Improving Throughput and Latency (at Different Layers)

Summarized by Muhammad Shahbaz (shahbaz@cc.gatech.edu)

Enabling Bit-by-Bit Backscatter Communication in Severe Energy Harvesting Environments

Pengyu Zhang and Deepak Ganesan, University of Massachusetts Amherst

Zhang started the talk by giving a short description of their new design for backscatter systems, which operates in severe energy harvesting conditions. He then talked about the current trends in lower-power sensors, specifically, the micro-powered sensors. These micro-powered sensors are equipped with energy harvesters and don't need batteries to power the whole system, which forces them to operate at very low energy. That's why these sensors are finding applications, specifically in the biosensing domain, such as delivering glucose in the blood stream and measuring vital signs using wearable sensors such as Google contact lenses. The physical limitations of these sensors makes it hard to attach batteries and thus designers are trying to find ways of using ambient and external energy sources to power the system—for example, harvesting energy from solar cells and from wireless signals.

Zhang et al. were interested in answering the question of how these devices interact with the outside world and what are the limiting factors. They looked into the example of RF harvesting and backscatter communication and found that the maximum operating range of these systems is 3.6 feet, or five times less than their communicating range of around 18.6 feet. This is because the atomic unit of existing network stacks didn't fit into the single discharge cycle and, thus, was limiting the overall operating range of the system. Using 1-bit as an atomic unit, they were able to achieve an operating range of 18 feet, but in that case throughput suffered at close distances. Later in the talk, Zhang gave different insights into maximizing communication throughput while maintaining the capability to operate at maximum range. By carefully selecting sleep time, the number of TX bits, and interleaving sensors on the packet level using a novel packet fragmentation and transmission scheme, they were able to achieve a 3.5 times increase in the operating range and a 5.8 times improvement in the overall throughput compared to the Dewdrop system.

Joshua Smith (a WISP designer) asked if the data doesn't come through, how will retransmission interact with the fragmentation scheme? Zhang argued that their scheme fragments the packets including the CRC into small microframes and performs all the error correction and detection once the packet is reassembled at the receiving end using the existing methods. In reply to Josh's follow-up question about the cost of the fragmentation scheme, Zhang answered that the packet is encoded into a fixed structure with a particular form, so they used trailing and leading bits to identify where packets began and ended. This adds some cost to the fragmentation scheme.

Full Duplex MIMO Radios

Dinesh Bharadia and Sachin Katti, Stanford University

Bharadia started the talk with a refresher on why full duplex radios have been considered an impossibility. He gave an example of two radios where the sending radio cannot receive the weaker signal because its own transmitting signal is acting as strong self-interference, which drowns the incoming signal. He said that it's analogous to hearing a whisper when one is shouting at the top of one's lungs. He later stated that the recent work in their lab, presented in last year's SigComm, has invalidated this assumption, and they have been able to demonstrate a fully working single antenna full duplex radio.

Before jumping into the details of their full duplex MIMO (multiple-input and multiple-output) design, Bharadia raised a question of why full duplex matters. Is it just the doubling of speed? He answered himself saying that the current performance curve is flattening, and we are running out of link layer techniques as well as reaching the channel limit. That's why full duplex is considered the next logical step and there is active interest from industry. Bharadia argued that for full duplex to be viable it needs to support MIMO. For full duplex MIMO radio, the design of an adaptive filter that matches the environmental reflections and cancels them should impose low complexity and minimal residue after cancellation.

Bharadia discussed the technical contributions of their work, where they designed a MIMO cancellation filter that has linear complexity in the number of antennas and doesn't scale quadratically compared to SISO replication-based design. In addition, the cancellation residue is ideal and doesn't degrade with the increasing number of antennas. In the end, he showed that their design does provide the proposed throughput scaling. The speedup is 1.95x, compared to the 35% improvement in SISO replication design, when matched against the standard half duplex design.

Shyam Gollakota (University of Washington and session chair) asked Zhang to compare the cost of WARP with the typical WiFi chipset, which by comparison is way more expensive than the WARP board's low cost. Gollakota also asked how the WARP components have greater linearity and the board has less noise compared to the typical chipset provided in mobile phones and access points. For the second question, Bharadia answered that linearity is typically based on standards. For example, WiFi needs 25 dB of maximum SNR (signal-to-noise ratio) so they design it at 30 dB and WARP also has around 30 dB of linearity. In his reply to the first question, he said that WARP may be $10,000 but the transceivers (Maxim chips) the board uses are fairly cheap, around $2 each. One can buy these off the shelf but this would require a lot of integration effort. He said that his point was when you buy these cheap transceivers they don't optimize linearity beyond a certain range. Thus, you have to cancel a lot of nonlinearities and noise. This requires building an analog design rather than a digital design because you cannot model noise in the digital domain.

Recursively Cautious Congestion Control (RC3)

Radhika Mittal, Justine Sherry, and Sylvia Ratnasamy, University of California, Berkeley; Scott Shenker, University of California, Berkeley, and International Computer Science Institute

Radhika Mittal started her talk by presenting a different view of the congestion control problem and how it's still unsolved. She mentioned the importance of classic work on congestion control by Van Jacobson and others and how it has helped in avoiding congestion collapse in the networks. But, she argued, this is not the only goal of congestion control, and users actually respond to fast completion time, which indirectly results in better revenue and profit.

Radhika presented a really interesting scenario: David Clark is a service provider providing services to the sender, Van Jacobson, and receiver, Vint Cerf. She showed how both resources (provisioned by the service provider) and link capacity are underutilized because of the current congestion control schemes. These schemes try to find a sweet spot between two conflicting goals of maximizing the throughput without adversely affecting other flows. RC3 tries to decouple these goals using priorities. RC3 utilizes spare capacity by sending additional packets using low priorities. This reduces the flow completion time and increases utilization of the bandwidth resources. She showed that assigning priorities required minimal changes in the TCP stack. Two parallel control loops, one for the regular TCP and the other for sending packets with multiple levels of priorities, were used to achieve max-min fairness.

Simulation results showed that RC3 performed 43.54% and 74.35% better on average over flows and bytes, respectively, compared to regular TCP. RC3 also showed better gains in completion time for short flows than existing schemes like RCP. Although better than regular TCP, the unoptimized RC3 implementation on Linux behaved relatively poorly compared to the simulated RC3 results, because the low priority packets were being processed by the slow path, thus causing high CPU overhead. The authors showed that leveraging NIC offloading capabilities like TSO and LRO reduced the overhead by half.

They also listed some cases where RC3 provides smaller benefits: for example, low delay bandwidth product and heavily utilized link, and some deployment concerns like partial support for priorities in switches. Looking toward the future, performance will eventually improve with the increasing bandwidth and round-trip time (RTT) product. The authors forecast a 45% and 66% reduction in average flow completion time over flows and bytes, respectively, in the futuristic datacenter with 100 Gbps bandwidth.

Dongsu Han (KAIST) asked about fairness among low priority packets. Radhika answered that this is ensured using the recursive multi-level priority scheme. Han asked whether there is an upper limit to the number of priority levels needed. Radhika replied that because they are using an exponential increase in the amount of packets sent per priority level, eight levels, for example, should be able to handle large flows with terabytes of data. Shyamnath Gollakota asked whether increasing the size of the buffer would have any effect on overall performance and delay. Radhika argued that it won't have much effect but will require more memory, but because memory is getting cheaper, this might not be an issue.

How Speedy Is SPDY?

Xiao Sophia Wang, Aruna Balasubramanian, Arvind Krishnamurthy, and David Wetherall, University of Washington

Xiao Wang gave a brief history of HTTP 1.1 and its global usage and mentioned, at the same time, that HTTP is not working as fast as desirable due to the increasing complexity of Web pages. Around 2009, Google introduced a new protocol called SPDY to solve many of the concerns in HTTP: e.g., opening a single TCP connection vs. opening individual connections for each object; the client's ability to prioritize objects; the server's ability to initiate requests, thus, avoiding extra RTTs; and compressing page headers along with the data. SPDY is now deployed in Chrome, Firefox, and on many Web sites. It's also the basis of HTTP 2.0, which is currently under standardization.

After briefly discussing the advantages of SPDY over HTTP, Wang listed various challenges, like variance in page load times and dependencies between network and browser computation. She then stated the goals of their paper: to perform a systematic study of SPDY and identify the dominant factors that are causing variability in its performance. The authors' approach was to extensively sweep the parameter space (e.g., network parameters like RTT and BW, TCP settings, and Web page effects) and isolate the dominant factors. Their finding was that SPDY helps on small objects because SPDY can batch up small objects into a TCP segment; SPDY helps on large objects under low packet loss due to reduced retransmissions from a single flow; and SPDY hurts on large objects under high packet loss because it backs off the TCP congestion window more aggressively than HTTP. Wang also found that object number, object size, and packet-loss rate are stronger indicators of SPDY's performance than RTT, BW, and TCP's initial cwnd. Their tests on real objects, although ignoring browser effects, showed that SPDY helped a lot, mainly due to its single TCP connection.

To capture the effects of the browser without worrying about variability in page load times, the authors introduced a new tool called Epload. They found that SPDY helps marginally when browser effects are preserved and suggested that the performance impact decreased due to browser computation and dependencies in real pages. For further improvement, the page load process needs to be restructured, for example with server push. Wang mentioned that people can download WProf and Epload at http://wprof.cs.washington.edu/spdy.

Someone asked about considering the effect of caching at the client or network or both. Wang answered that caches present similar behavior as having a small number of objects. If the number of objects is very large, SPDY actually helps less, but they don't consider the caching case in the network. James Mickens (MSR) started his question by saying that all browsers are terrible, which made the audience laugh. He said that they not only differ in their computation but also along other axes like storage and the network stack they use. He asked how the emulator, Epload, generalizes across all these different browsers. Wang explained that Epload is based on WProf, the work they presented in last year's NSDI. Using WProf, Epload captures these dependencies in a browser-independent manner. She said that they ran test cases under different browsers in order to capture dependencies across browsers. Wang did mention that the emulator cannot capture the computation time variability, but based on the platform, they can hypothetically increase or decrease the compute time.

In-Memory Computing and Caching

Summarized by Qiao Zhang (qiao@cs.washington.edu)

FaRM: Fast Remote Memory

Aleksandar Dragojević, Dushyanth Narayanan, Orion Hodson, and Miguel Castro; Microsoft Research

Aleksandar Dragojević started by discussing two hardware trends that make FaRM timely and important. He noted that it is becoming cost-effective to store almost all application data in memory. While new datacenter networks promise larger throughput and lower latency, network communication remains a bottleneck for systems that use TCP/IP. Fortunately, RDMA technology that allows direct read/write of remote memory is now available at competitive prices. To take advantage of the performance gains from kernel bypassing, FaRM builds a message passing primitive using fast RDMA writes. Microbenchmark for remote random reads between RDMA, RDMA-based messaging, and TCP shows that FaRM's RDMA-based messaging can achieve an order of magnitude improvement on both throughput and latency compared to TCP.

FaRM explores how to use RDMA to build distributed systems. Aleksandar explained that in order to program a modern cluster with terabytes of memory, hundreds of CPUs, and RDMA network, we want to keep data in memory, access data using RDMA as much as possible, and co-locate data and computation because accessing data locally is a factor of 20 faster than accessing remotely even using RDMA. Moreover, RDMA lends itself to a symmetric model where machines not only store data but also execute applications in order to exploit data locality and to avoid idle server CPUs.

FaRM simplifies programming by providing a shared address space and general distributed transactions with strong consistency guarantees. FaRM allows applications to read/write and create/free objects in a shared address space using ACID transactions. FaRM supports locality-aware optimizations that allow new objects to be created closer to existing objects to exploit locality. For performance-critical operations, FaRM allows applications to perform an efficient lock-free read in a single RDMA operation by taking advantage of cache-coherent DMA.

Aleksandar presented the implementation of a distributed key-value store and a Tao-like in-memory graph store on top of the FaRM APIs. The FaRM key-value store achieves 16x the throughput and two orders of magnitude improvement in latency compared to the state-of-the-art TCP-based key-value store. The FaRM graph store achieves 10x the throughput and a 40x–50x improvement in latency compared to the TCP-based in-memory graph store.

Someone asked whether it would be more accurate to compare a FaRM messaging primitive to UDP since RMDA requires lossless networks. Aleksandar replied that UDP can achieve lower latency but it is nowhere close to RDMA. Hein Meling (University of Stavanger) asked whether the next step for FaRM is to support across-datacenter applications. Aleksandar said that the latency between datacenters would be too high. The next step is to explore the right primitives to put in the NIC to further improve performance and enable new functionality, and also to implement other applications using FaRM. Someone asked what the difference was between FaRM and RAMCloud. Aleksandar answered that the main difference stems from motivation: RAMCloud is a key-value store, whereas FaRM exposes a much richer and more general programming model with shared address space and distributed transactions. Jeff Rasley (Brown) asked why the evaluations are done using only 20 machines and whether that signals scalability limitations with RDMA. Aleksandar replied that they only had 20 machines and commented that FaRM can scale to 100 machines. Wang (Cisco) asked whether FaRM migrates data to exploit locality. Aleksandar replied that FaRM does not migrate data but keeps data and computations together, and further commented that they expect FaRM to run a single large application, so there would be no contention.

Easy Freshness with Pequod Cache Joins

Bryan Kate, Eddie Kohler, and Michael S. Kester, Harvard University; Neha Narula, Yandong Mao, and Robert Morris, MIT/CSAIL

Bryan Kate started his talk by arguing that application caches should support materialized views natively because in-cache materialized views are easy to use and have good performance. Existing application caches, like Memcached and Redis, provide fast key-value cache to offload reads from database, but the burden of maintenance rests on applications to keep the cache fresh. As a motivating example, Bryan explained how the Twitter timeline is constructed from a join between subscription lists and user data, and pointed out that the result of the join should be cached because the timeline is checked very frequently. While it is easy to cache the join, it is difficult or cumbersome to update the cache when there are new posts. A simple solution is to use a materialized view supported by modern database systems that compute, store, and automatically update the query results. However, the database is often designed for durable storage and therefore becomes a performance bottleneck when tasked to handle frequent reads and writes.

In order to help applications to avoid the complexity of keeping the cache fresh and to provide good performance, Bryan presented Pequod, a distributed application cache that provides materialized views in a key-value cache with operations such as get, put, scan, and join. Bryan introduced the key idea of Pequod cache joins that allow applications to relate computed data (e.g., timeline) to base data (e.g., posts and subscriptions). Pequod also offers a number of advanced features such as partial and dynamic materialized views, incremental updates, and eager or lazy updates. Bryan highlighted that distributed Pequod can scale to handle large data sets. Computation is kept local while base data is partitioned and transparently replicated when necessary to allow cache joins to be computed anywhere. Finally, Pequod supports cache eviction under memory pressure and allows for eventual consistency.

Pequod was evaluated on a Twitter-like benchmark, including timeline checks, new subscriptions, and new posts. The first experiment compared Pequod with fast key-value caches, such as Memcached and Redis, and a DB-as-cache database, such as Postgres. Results showed that Pequod performed no worse than existing caches. The second experiment tested how Pequod performance scaled with additional servers. Results showed a 3x increase in performance when the number of servers increased from 12 to 48. The imperfect scaling resulted from data movement between servers. The overhead was noticeable but not crippling. Bryan mentioned related work, e.g., DMV, DBProxy, and PNUTS.

Someone asked about latency in Pequod and how latency is affected when data are evicted under memory pressure since Twitter-like applications expect low latency. Bryan replied that there is a latency spike the first time a cache join is computed, and that applications can scale out to hold a larger cache to minimize eviction and avoid high latency. Moreover, Pequod allows tunable eviction of computed data vs. base data. A further question concerned eventual consistency in Pequod when Twitter users expect up-to-date results. Bryan answered that application developers have grown used to eventual consistency, and in Twitter, writes may take up to a second to trickle to user timelines.

MICA: A Holistic Approach to Fast In-Memory Key-Value Storage

Hyeontaek Lim, Carnegie Mellon University; Dongsu Han, Korea Advanced Institute of Science and Technology (KAIST); David G. Andersen, Carnegie Mellon University; Michael Kaminsky, Intel Labs

Hyeontaek Lim presented MICA, a fast in-memory key-value store. MICA aims to improve per-node performance and mainly targets small k-v items that can fit in single packets. Hyeontaek compared the end-to-end performance over the network on the

YCSB workload between MICA and current systems (e.g., Memcached, RAMCloud, MemC3, Masstree) using a single server node. While the performance of the current systems collapses under a write-intensive workload, MICA achieves orders of magnitude improvement for both uniform and skewed workload and attains close to maximum packets/sec possible using UDP.

Hyeontaek explained that the significant improvement comes from MICA's approach, which applies new software architecture and data structures to general-purpose hardware in a holistic way. He outlined three key design choices: First, to avoid frequent cache line transfers between cores and to allow CPU performance to scale with the number of cores, MICA partitions data and gives each core exclusive access to a partition. Experiments show that exclusive access outperforms concurrent access in MICA. Second, MICA uses client-assisted NIC-based request direction to ensure correct and high-throughput delivery of requests to the respective cores for exclusive access. Third, MICA proposes a new "cache" data structure that uses a circular log and lossy concurrent hash index for each partition to provide high throughput for both reads and writes. Hyeontaek emphasized that these unconventional design choices allow MICA to achieve good performance for both throughput and latency, and for both uniform and skewed workload, compared to current systems. Source code can be found at github.com/efficient/mica.

Someone asked how much each of the three tricks contributes to the performance gain. Hyeontaek replied that the paper contains experimental results that show performance comparisons between each design choice and its alternative, e.g., concurrent and exclusive access. Someone from the Voldemort project commented that the performance gain might be much smaller, taking into account the management overhead once MICA becomes fully featured, because it is not fair to compare a research prototype with a fully featured product like Redis or Memcached. Someone from UCSD asked how MICA compares to partitioned k-v stores such as single-threaded Memcached in each core. Hyeontaek answered that such deployment can reproduce most of the gains from MICA design. However, Hyeontaek pointed out that MICA has a hybrid mode that allows both concurrent and exclusive access that is only possible in MICA architecture. Ryan Stutsman (Microsoft Research) asked how MICA keeps track of where cache misses are coming from since circular logs can evict data and result in dangling references. Hyeontaek answered that MICA does not do dynamic reconfiguration of the system. As a solution, one can use lossless data structure or try to predict how much of the losses are coming from data structures.

Scalable Networking

Summarized by Feng Lu (f1lu@cs.ucsd.edu)

NetVM: High Performance and Flexible Networking Using Virtualization on Commodity Platforms

Jinho Hwang, The George Washington University; K. K. Ramakrishnan, Rutgers University; Timothy Wood, The George Washington University

Jinho Hwang began by reviewing recent developments in high performance networking such as PacketShader, Intel DPDK, etc. When the datacenter is virtualized, the performance of these systems could suffer due to the virtualization overhead. He then turned his focus to a specific domain—network function virtualization—and identified two potential challenges: high speed/low latency processing and efficient inter-function (VM) communication. Both challenges motivated his work, NetVM, which aims to provide complex network functionality at line rate (10 Gbps) using commodity hardware (e.g., Intel DPDK and commercial off-the-shelf servers). Jinho discussed two possible placements of DPDK in a virtualized environment: in the hypervisor and in the VM with on-NIC L2 switch. He pointed out that neither could achieve a full line-rate network speed in VMs.

Having explained the limitation, Jinho described the overall design of NetVM, which allows memory sharing between hypervisor and VM, and among VMs themselves. Subsequently, he sketched out the design challenges faced by NetVM: how to ensure zero copy, huge-page sharing between the VM and the hypervisor, lockless and NUMA-aware design, and security domains. He then talked about how NetVM addressed each challenge in turn. For zero-copy, NetVM directly DMAed packets into shared memory and only packet control structures are passed around. For sequential packet processing, packet references are passed between VMs. At any time, only one VM can process the packet to limit concurrency. For lockless and NUMA-aware design, NetVM has dedicated core-queue matching, data-structure separation, and processing path alignment in both the hypervisor and the VMs. For huge-page virtual address mapping, pre-calculation was used to speed up offset lookup. Finally, trusted VMs were using shared memory in NetVM while untrusted VMs see packets via the hypervisor.

Afterward, Jinho moved to the evaluation section and compared NetVM to SR-IOV-VM, Click_NetVM, and Click_Native_Linux. NetVM was able to attain packet delivery and forwarding rate at full line speed, while second best Click_NetVM, was only able to achieve 6 and 6.8 Gbps, respectively. For inter-VM forwarding evaluation, due to the limited number of cores, NetVM could only sustain a line rate of up to three VMs. However, Jinho also mentioned that for a more realistic workload (60% partial forwarding), NetVM would maintain a 10 Gbps line rate up to five VMs.

Wang (Cisco) said that it is common to pass packets in and out of applications, but in reality, people are used to socket interface; his question was about the programming model. Wang said that Jinho mentioned netmap; the beauty of netmap is that it also supports a socket-like interface. Jinho restated the question as a protocol stack issue, mentioning that NetVM was targeted for middlebox applications, which handle packets. However, there was some software running within applications, such as mTCP, that provided a user-level network protocol within it. In most cases, people use a customized user-level network stack. Wang then asked how much code needed to be changed. Jinho mentioned that KVM was modified to achieve queue/core alignment and to add a PCIe device. They didn't touch any kernel module and only one kernel module was introduced to support user space I/O. Finally, Wang asked whether they dedicated one interface for one VM or could traffic from one interface be shared with multiple VMs? Jinho replied that NetVM allowed one interface for multiple VMs. But to maximize performance, they dedicated one interface to one VM.

ClickOS and the Art of Network Function Virtualization

Joao Martins and Mohamed Ahmed, NEC Europe Ltd.; Costin Raiciu and Vladimir Olteanu, University Politehnica of Bucharest; Michio Honda, Roberto Bifulco, and Felipe Huici, NEC Europe Ltd.

Joao began by pointing out the various drawbacks associated with hardware middleboxes: price, power, management, scalability, and so on. Given these limitations, Joao suggested pushing middlebox functionality to software for the following benefits: shared hardware across multiple tenants, reduced equipment/power costs through consolidation, and flexibility to try out new features. He then articulated the problem solved by ClickOS: Middleboxes can be built on commodity hardware while still achieving high performance. He briefly went over the achievements made by ClickOS: fast boot (30 ms), small memory footprint (5 MB), isolation by Xen, 10 Gbps line rate processing, and flexibility based on the Click library.

Joao continued the talk with more details on ClickOS architecture: Instead of a traditional guest OS on Xen, ClickOS built on top of MiniOS with a single application on a single core. In addition, the Click control plane was emulated over MiniOS/Xen and the boot time was reduced to 30 ms (uncompressing the VM takes longer than booting). He also mentioned that various optimization tricks were used to enable line rates of 10 Gbps. Joao next moved to performance analysis with a native implementation of ClickOS and identified three pieces on the packet processing pipeline: open vSwitch, netback, and netfront. Unfortunately, the performance of these modules was far behind line rate. He quoted that 14.88 million packets would be processed to sustain line rate given a 64-byte packet size. However, these modules can only process 250k–350k packets per second. He attributed the poor performance to packet copy between Click/Xen (772 ns), packet metadata allocation (600 ns), and a slow back-end switch. To address these problems, several optimization techniques were used: reuse Xen page permission, replace OVS with Vale switch, increase I/O batch size, and use the netmap API all the way to the NIC buffer (kernel bypass).

He then presented an overview of the ClickOS prototype and emphasized that the Click changes were small. He explained the evaluation setup: Intel Xeon with Sandy bridge, 16 GB RAM, one Intel NIC and one CPU core for VMs, and the remaining cores dedicated to dom0. He navigated through the evaluation section, started with base transmission performance while varying the TX ring size. With ring size >= 1024, ClickOS could achieve line rate at all packet sizes. Next, the virtualized middlebox performance was evaluated in a three node setup (host1—>ClickOS—>host2) for a wide range of middlebox functionalities. Finally, the effectiveness of their Linux optimization was presented. He also mentioned that the entire source code is now available at: http://cnp.neclab.eu. Joao concluded his talk with future work, which aimed to consolidate thousands of VMs, improve inter-VM communication, and exploit boot times to achieve reactive VMs.

Marv Ayre (Princeton) wondered about CPU load and utilization among the cores. Joao replied that there was just one core to handle the NIC if it receives packets on the virtual port—in short, just one core in dom0. For guest OS usage, if it is not receiving the packet, the CPU usage is zero. The guest OS did not use anything like DPDK, and there was no polling. Ayre then asked whether they changed any code to handle high load. Joao said their design handled high load well and referred people to the paper for the results. They booted 100 VM instances, and these instances could fill up the pipe. Someone asked about the 100 VMs example: Was anything done about the CPU scheduling or about latency? Joao replied that for 100 VMs and beyond there is likely a bit of a scheduling issue, but he emphasized that the VMs were scheduled fairly. Although the individual contribution per VM rate decreased slightly, overall they did not see any issue. Someone else asked whether they used Xen's virtual IRQ to trigger interrupt. Joao answered yes and confirmed that they used the event channel to deliver interrupt. The same person asked whether they used polling, and Joao replied that since the interrupt happens on a per-batch basis, constant interrupts aren't occurring. In addition, the guest OS does not poll the back end.

SENIC: Scalable NIC for End-Host Rate Limiting

Sivasankar Radhakrishnan, University of California, San Diego; Yilong Geng and Vimalkumar Jeyakumar, Stanford University; Abdul Kabbani, Google Inc.; George Porter, University of California, San Diego; Amin Vahdat, Google Inc. and University of California, San Diego

Siva began his talk by discussing server consolidation, multitenancy, and network resource management and sharing in datacenters. He further identified a key piece of functionality, namely a programmable rate limiter on the end host to ensure performance isolation and effective congestion control. He then discussed two options for rate limiters in existing systems. Software rate limiters are scalable but not accurate and precise. Hardware rate limiters (available on NICs) could easily achieve the latter two but are not scalable. The authors propose SENIC as a way to combine the advantages of both approaches by reorganizing the responsibility of the operation system and the NIC functionalities.

Siva reviewed the current NIC design and identified that existing NICs unnecessarily pre-DMAed packets from host memory to NIC buffers, which are limited in size and not able to scale up. Instead, Siva proposed per-class queues in host memory and having the NIC compute the schedule on demand for each packet. The late binding of packet transfer to NIC enables accurate and precise rate limiting in SENIC. He continued with SENIC implementation details and discussed how they implemented SENIC on both software (as a Linux kernel module) and hardware (NetFPGA). For the software prototype, a dedicated CPU core is used for network scheduling, and for the hardware prototype, token bucket scheduling is implemented.

Siva reviewed microbenchmark results on 10G with NetFPGA. Their current NetFPGA prototype supports 1000 rate limit classes. The inter-packet delay for a traffic class was studied, and the authors found that the average and standard deviation was within 0.038% and 1.7% of ground truth, respectively. Siva next discussed scheduling latency. It takes SENIC 50 ns to compute the schedule for the next packet, which is well within the allowed budget (300 ns for 1500 bytes for 40G). In the last part of the evaluation, Siva talked about tenant isolation by configuring 10 memcached tenants (6 Gbps) sharing the network with one UDP tenant (3 Gbps). He presented the 99.9th percentile tail latency of memcached tenants while varying the number of requests. The tail latency was below 5 ms even when the aggregated load approached 6 Gbps. Additionally, the UDP client was able to attain 3 Gbps irrespective of memcached workload. Both results significantly outperformed the two existing approaches, namely HTB and PTB, provided by the current Linux kernel. Siva explained how SENIC supported other NIC features and chose TSO as an example. Source code for SENIC is available at http://sivasankar.me/senic.

Someone wondered how easy it is to implement their solution on commodity NICs. Siva replied that most of the parts used in SENIC, such as packet scheduler and DMA engine, are already available on the NIC. Packets are in the host memory and DMAed into the internal ring buffer. What is needed is a scalable scheduler to compute scheduling on demand and pull the packet for transmission.

mTCP: A Highly Scalable User-Level TCP Stack for Multicore Systems

EunYoung Jeong, Shinae Woo, Muhammad Jamshed, and Haewon Jeong, Korea Advanced Institute of Science and Technology (KAIST); Sunghwan Ihm, Princeton University; Dongsu Han and KyoungSoo Park, Korea Advanced Institute of Science and Technology (KAIST)

Awarded NSDI '14 Community Award!

Shinae Woo motivated mTCP by arguing the need to handle a large number of short flows and the high cost of connection management in datacenter networks. She explained why the current TCP implementation in Linux is unsatisfactory: in particular, the kernel is not well designed to sustain line rate with small flows and does not scale well with respect to the number of cores. She then presented a Web server example, where the CPU spent more than 80% of time in kernel with 34% spent in the TCP/IP stack. She attributed the performance bottleneck to shared resource contention, broken locality, and per-packet processing overhead in the Linux kernel. She then discussed several related works in addressing the kernel inefficiency and pointed out that none of them solved all of the aforementioned problems.

mTCP is a clean-slate design for a user-level TCP implementation. Woo emphasized that mTCP is explicitly designed for multicore systems and listed mTCP design features, such as independent cores with no resource sharing, resource affinity, batch packet processing, and a portable API compatible with the current Linux kernel. In particular, she mentioned that each mTCP thread corresponds to one application thread and that mTCP threads extend the PSIO library to support efficient packet I/O interface with lock-free and per-core data structures, and she addressed the context switch overhead. Additionally, she talked about porting existing apps on mTCP; most apps required less than 100 lines of changes. She continued with some implementation details, such as 11,473 LOC for mTCP, 552 lines to patch the PSIO library, and that mTCP follows RFC 793.

Woo cited some microbenchmark results and showed that mTCP could scale linearly with CPU cores when handling 64-byte packet connections. She then discussed the performance improvements of ported application on mTCP versus Linux and MegaPipe, using two application examples: lighttpd and SSLShader. With lighttpd, mTCP improved over Linux and MegaPipe by 3.2x and 1.5x, respectively. With SSLShader, where one-byte objects were downloaded, mTCP boosted performance by 18% to 33% over Linux. The source code for mTCP is available at: http://github.com/eunyoung14/mtcp.

Someone asked whether they used timers to batch I/O. Woo responded that they did not use any timers to batch threads. This naturally happens with the context switching. The next questioner pointed out that from netmap they've learned that allocating packet structures such as sk_buff is expensive and that, in this work, scheduling is killing performance, maybe system call overhead as well. What was the comparative overhead given that they measured the whole kernel, that is, where were the performance gains coming from? Shinae answered that they were not aware of exactly where the performance gains come from, such as which part contributed most benefit. Context switching is generally more expensive than system calls. Shinae stressed that they batched 2000 events per context switch, and it seemed that batching provided more benefit. A third questioner wondered whether they could move the scheduling portion of mTCP into the kernel. Shinae replied that it is hard for the kernel to provide an event-driven API to applications since the two communicate via system calls. Applications would have to be changed to support an event-driven model. In mTCP, applications do not need to be changed and they still use the typical connect() and send() calls. However, the batching is provided by the mTCP stack. Shinae believed it would be hard to support batching in kernel. The final questioner noted that they had divided the application into two parts so that applications were aligned to cores; the questioner wondered whether they had to rewrite the application so that they had a process running per core. Shinae replied that the ported applications normally support a single-process multiple-thread model, in which they can easily change the API to use enough cores. For a single-process single-thread model, they would have to make changes to turn it into a single-process multiple thread.

New Programming Abstractions

Summarized by Oliver Michel (oliver.michel@colorado.edu)

Warranties for Faster Strong Consistency

Jed Liu, Tom Magrino, Owen Arden, Michael D. George, and Andrew C. Myers, Cornell University

Jed Liu started with a discussion about the tradeoff between consistency and scalability in distributed systems. He illustrated this tradeoff by comparing relational database systems and Web-scale NoSQL systems only providing weak, eventual consistency. Essentially, these systems are harder to program against because consistency failures are likely.

To bridge this gap, the authors introduce warranties, which are time-limited assertions about the state of a system. State warranties ensure that a certain key has a specific value until some point in time, whereas computation warranties guarantee that a certain computation result holds true until some point in time. For example, such a warranty could assert that there are at least x seats available on a flight in an airline booking system until a specified time. Warranties are strictly serializable and provide a generalized form of optimistic concurrency control. They provide improved scalability and throughput in read-heavy environments, whereas write access may become a bottleneck because writes that would violate assertions are delayed until the warranty expires.

The authors' evaluation showed that a key parameter, heavily influencing the system performance, is the duration of a warranty. Thus, these durations must be chosen carefully. Warranties must be valid long enough to be useful but short enough to keep the system from blocking access completely. Finally, Liu provided performance figures showing a speedup of 2x is achieved by warranties for a system with only 2% of data access being writes. Performance worsens in comparison to not using warranties at about 9%–10% writes.

Aaron Gember (University of Wisconsin) asked how long warranties typically are. Jed answered that warranties are in the range of a few seconds depending on performance characteristics of the system. Someone from Microsoft Research asked whether a detailed knowledge of the performance characteristics of the system is needed to set durations for computational characteristics appropriately. The author answered that a good knowledge of the workload is necessary. Eddie Kohler (Harvard University) asked whether some warranties are more useful than others. Jed answered that this is certainly true and some messages are not even worth having against the warranty overhead. Alec Wolman (Microsoft Research) asked how developers would tune the system if the workload changed over time. Jed replied that they haven't thought deeply about this, but would have to think about which methods get memoized. Aditya Akella (University of Wisconsin-Madison and session chair) asked whether the higher throughput resulted in higher latency. Jed said that the latency was just for write transactions, and for a read-mostly workload, this is acceptable.

Tierless Programming and Reasoning for Software-Defined Networks

Tim Nelson, Andrew D. Ferguson, Michael J. G. Scheer, and Shriram Krishnamurthi, Brown University

Tim Nelson opened by discussing the three main application tiers in programming software-defined networks: the flow-rules in the switches, the controller program, and the store for the controller state. In most languages, these three layers are abstracted differently. In fact, there is a gap between the code that the controller application executes to produce a new flow modification message and the interface that receives this information on the switch side, which leads to errors and inconsistencies between the tiers.

Tim introduced Flowlog, the authors' approach to a completely tierless programming environment for software-defined networks. Flowlog provides a cross-tier interface such that applications written in this language are easier to statically verify since the gaps between the tiers are covered in the language runtime directly. This runtime handles compilation to flow tables while including state and state updates. With this approach (where also all flows are proactively pushed to the switches), bugs can be found before the rules are executed on the switch. For verification, standard utilities like Alloy can be used because Flowlog programs are equivalent to first-order logic programs.

Aaron Gember (University of Wisconsin) asked whether the authors thought about supporting middlebox programming in their system, which they did not. Concerns were raised that this abstraction (with a SQL-like language) is too high-level; how would languages like Erlang fit into this project? In fact, the authors really thought about using a functional language instead of a custom DSL. Someone asked whether they could still run into inconsistencies between layers. Tim replied that they don't allow state to expire due to lack of use. They are adding the ability to have timeout events for removing rules. Aditya Akella (University of Wisconsin–Madison) asked about how the compiler works and whether it can handle all types of consistency semantics. Tim replied that Flowlog supports some things that OpenFlow does not. When their compiler detects forwarding rules, it produces netcore code and uses the netcore compiler. Andrew Ferguson (another author) pointed out that some parts of the policy might be directed toward a single switch, and that that is an abstraction introduced by netcore.

Enforcing Network-Wide Policies in the Presence of Dynamic Middlebox Actions using FlowTags

Seyed Kaveh Fayazbakhsh, Carnegie Mellon University; Luis Chiang, Deutsche Telekom Labs; Vyas Sekar, Carnegie Mellon University; Minlan Yu, University of Southern California; Jeffrey C. Mogul, Google

Seyed Kaveh Fayazbakhsh gave a short introduction to SDN and middleboxes and pointed out that integration of middleboxes in software-defined networks is a non-trivial task since middleboxes modify packets based on traffic dynamics. This behavior violates two key tenets of software-defined networking, namely origin binding and the paths follow policy. Origin binding means that there should be a strong binding between a packet and its origin, which is violated by NAT boxes. The paths follow policy states that explicit policies should determine the paths that packets follow, which Web proxy boxes violate. As a result, management, debugging, and forensics are more complicated, and, furthermore, it is difficult to ensure service-chaining policies.

Seyed explained that to solve this issue, it is necessary to add the missing contextual information that middleboxes otherwise may rewrite. The authors achieve this goal by adding tags to flows, where a central flow tag controller configures the tagging logic. Subsequently, critical information that middleboxes need is encoded in tags, and forwarding is performed based on tags not overwriting information like the source IP address. Through this central configuration, enhanced policies are possible and are expressed in dynamic policy graphs. These graphs hold transition conditions between middlebox nodes from sources to destinations, which makes chaining devices more easily feasible.

To implement this system, middleboxes need to be modified. This is possible because middleboxes already rewrite packets at a high level. That means that actually no changes to their internal logic are needed. The code to add in a middlebox is negligible (25–75 LOC). The processing overhead is less than 1%. Based on their evaluation, only 15 bits are needed to encode tags, which can be done in, for example, the IP-ID or the IPv6 flow label. Additionally, flow labels add enhanced semantics to a packet, which allows for extended analysis and debugging.

Questions were raised about how granular flow tags need to be. In the worst case, every flow tag corresponds to a transport layer flow. The flow tags also caused confusion around how exactly flow tags are possible in OpenFlow. Seyed said that because OpenFlow and most middleboxes support rewriting IP-ID and other possible headers, flow tags can easily be implemented, and that 15 bits for flow tags supports 70k-80k flows per second.

Closing Remarks

Summarized by Rik Farrow

USENIX Executive Director Casey Henderson closed the symposium by thanking the chairs and handing a bottle of sparkling wine to Ratul. Paul Barham (Microsoft Research) and Arvind Krishnamurty (University of Washington) will be the co-chairs for NSDI '15.

LISA14

Nov. 9–14, 2014 | Seattle

More Craft. Less Cruft.

Wednesday Keynote Speaker:
Ken Patchett, Director of Data Center Operations, Western region, Facebook

Thursday Keynote Speaker:
Gene Kim, former CTO and founder, Tripwire, co-author of *The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win*

Closing Plenary:
Janet Vertesi, Princeton University

Featuring talks and training from:
Michael “Mikey” Dickerson
Caskey Dickson, Google
Garrett Honeycutt, LearnPuppet.com
Dinah McNutt, Google
Laura Thomson, Mozilla
James Turnbull, Docker
Avleen Vig, Etsy
Mandi Walls, Chef

Full Program and Registration Coming August 2014

www.usenix.org/lisa14

usenix