Hot-ICE '11: Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services

Boston, MA
March 29, 2011

Cloud Resource Management

Summarized by Theophilus Benson (tbenson@cs.wisc.edu)

Dynamic Resource Allocation for Spot Markets in Clouds

Qi Zhang, Eren Gürses, and Raouf Boutaba, University of Waterloo; Jin Xiao, Pohang University of Science and Technology

Current cloud computing offers on-demand computing at fixed prices; however, these fixed pricing models lead to underutilization of the cloud. Some cloud providers have begun to adopt spot-pricing models: a spot market for each class of virtual machines. The number of physical hosts allocated to each market is fixed. Qi Zhang claims that static allocation of physical resources to each market restricts the flexibility of each market and reduces the maximum achievable revenue. To this end, Qi proposed a technique to dynamically adjust the supply of physical resources to reach market, with the sole goal of maximizing the amount of revenue received by the cloud provider.

There are two challenges to tackle in developing this technique. First, accurate prediction of the demand for each market. This is accomplished by using an auto-regressive (AR) model to predict future demand based on prior demand. Second, developing an algorithm to utilize the demand to allocate the resources to each market. The framework, proposed by Qi, consists of three components: a market analyzer, a capacity planner, and a virtual machine scheduler. The market analyzer uses the AR model to predict demand. The capacity planner uses the observation that the demand curve is a non-increasing function of spot price to determine the spot prices for the different markets based on the predicted demand. Given the spot price, the capacity planner optimizes for the product of price and quantity. The authors show that this concave optimization problem can be reduced to the multiple knapsack problem. Finally, the scheduler launches a virtual machine and adjusts the supply of resources to each market based on the output of the capacity planner.

To evaluate the framework, Qi conducted experiments on CloudSim by modifying the scheduler to incorporate his framework. In comparing static allocation of physical resources to dynamic allocation the authors showed that their framework improves the revenue of the cloud provider.

How does granularity of decision-making affect total price? Currently, the granularity of decision-making isn't evaluated, but it will be in the future. How does the constraint of fixing the size of the pools based on predicted demand affect the solution? The current system is designed to improve prices via increased flexibility. Future work will examine this.

On the Benefit of Virtualization: Strategies for Flexible Server Allocation

Dushyant Arora, Anja Feldmann, Gregor Schaffrath, and Stefan Schmid, Deutsche Telekom Laboratories/TU Berlin, Germany

Dushyant focused on the migration of services between different locations to improve users' QoS while reducing the economic cost associated with migrating the servers of the service. This work is prompted by the fact that users are mobile and over time their request patterns change. For example, countries will have different access patterns due to different time zones, and users' daily commute will cause them to access resources from various locations. The goal of this work is to provide worst-case guarantees without knowledge of future demand. To provide these guarantees, the authors chose to employ an online algorithm.

Their algorithm divides time up into epochs and at the end of each epoch it uses the request patterns seen to approximate where future request patterns will originate from. With this prediction, a server is picked and moved towards the center of gravity of its request patterns.

In evaluating the effectiveness of this algorithm, Dushyant compared his algorithm with an optimal offline algorithm that makes choices using actual future demands and with a static online algorithm. The algorithms are compared under two different patterns: time-of-day requests originating from different countries, and commuter requests originating from different locations. The authors show that their algorithm is beneficial in times of moderate dynamics, as it is able to exploit the request patterns.

Is each epoch static or migration-time static? Epochs are defined based on the latency between clients and servers. Do you assume a constant amount of time to get the monitoring information? Yes, there is a centralized infrastructure that gathers all the information. Currently, constant time is assumed. Certain organizations include policy such as middlebox interposition—how do you account for such policies? Currently, they do not account for such policies, as these policies complicate the algorithm; however, they address other policies. Furthermore, they don't assume intimate knowledge of provider topology.

Cost-Aware Live Migration of Services in the Cloud

David Breitgand, IBM, Haifa Research Lab; Gilad Kutiel and Danny Raz, Technion, Israel

Live migration consists of two phases: the pre-copy phase, when initial memory pages are copied to the new location while the virtual machine is running, and the copy phase, when the virtual machine is stopped and the un-copied or dirty pages are copied over. David claims that the pre-copy phase is equally important as the copy phase, as the processing power and bandwidth used during the pre-copy phase impacts the response time of the applications and could lead to SLA violations. David assumes that live migration is performed in band—the same network that is used to service requests is used for migration. In choosing the amount of bandwidth to reserve for migration, the operator faces a tradeoff, since reserving small bandwidth leads to longer migration, while reserving large bandwidth will reduce migration time but degrade the service.

The cost of migration is, then, the cost of the pre-copy phase and the cost of the copy phase, both of which are a function of the bandwidth reserved for migration. The authors propose an optimization algorithm for dynamically deciding the amount of bandwidth to reserve for the pre-copy phase, based on the number of pages to copy and the amount of time elapsed.

To evaluate their optimization, the authors use a trace-driven simulation and compared this algorithm to the default algorithm in Xen. Their experiments show that the framework is able to better adjust itself according to the available bandwidth and cost function than Xen's default algorithm.

To what extent is this solution affected by the assumption that bandwidth is a bottleneck because the same link is used for both migration and service requests? The solution applies in situations where bandwidth isn't a bottleneck. Do they assume time to migrate is not dependent on workload, as workload affects time of migration because workload may largely affect memory? The model contains provisions for determining the probability of modifying the memory pages and is able to simulate workloads with varying impact on the memory. Do they measure downtime from the server, when the server is turned off and moved, or from the client, when the user perceives disruption in service? From the client's perspective.

Server Operational Cost Optimization for Cloud Computing Service Providers over a Time Horizon

Haiyang Qian and Deep Medhi, University of Missouri—Kansas City

Haiyang claimed that servers account for a large portion of the operational costs of maintaining a datacenter. Further-

more, he claimed that the operational cost for servers can be broken down into energy consumed by the server and the cost of replacing hard disks. The authors' goal is to determine the optimal frequency and number of servers to use to satisfy demand for the various services while minimizing the cost of running the datacenter. To do this, the authors formulate an optimization problem that minimizes the power consumption, the cost of turning on and off servers, while ensuring the constraint that each server can only run at one frequency and that the datacenter must satisfy all the users' demands.

Haiyang showed that the optimization problem is quadratic in nature and very time-consuming, and thus he developed optimizations to make the problem linear. In evaluating his formulation, he compared the cost of maintaining a datacenter under different server allocation strategies. Haiyang observed that adjusting CPU frequency provided significant cost improvements over running the servers at max CPU frequency. Haiyang also noted that the granularity of time over which calculations are made impacts the cost; larger time granularity costs less.

Do they assume the load can be moved to other locations in the datacenter, and do different servers have access to the same resources? Yes, they do assume migration; however, they didn't look at other locations. Do people actually change the frequency of the CPU? Yes, certain operating systems come with sufficient functionality to change the CPU frequency. What is the impact of CPU throttling on the performance of the application, and how does throttling affect the ability of an application to meet its SLA? They assume all service level agreements are satisfied.

Service Management

Summarized by Theophilus Benson (tbenson@cs.wisc.edu)

Automated Incident Management for a Platform-as-a-Service Cloud

Soumitra (Ronnie) Sarkar, Ruchi Mahindru, Rafah A. Hosn, Norbert Vogl, and HariGovind V. Ramasamy, IBM T.J. Watson Research Center

Cloud providers offer tenants on-demand access to resources at low cost. To maintain this low cost while remaining competitive, providers are forced to reduce operational expenses by limiting real-time problem determination and eliminating SLA guarantees. The authors propose a framework to further reduce the amount of problem determination by using simplified automated corrective actions. The framework performs integrated monitoring for virtualization systems across all layers: networks, physical, hypervisor, and OS. The framework responds to issues by taking the automated action for some incidents and ticket creation for other items. In a small number of scenarios both actions are taken; this is to inform system admins of actions via ticket creation.

The framework is a centralized system which aggregates information from multiple sources, including agents installed on the physical hosts. In designing this system, the authors had the following design requirements: scalability, persistence, and fault tolerance. To achieve persistence, data is stored to persistent cloud storage. For scalability, each server is modeled using a simple finite state machine (FSM). Using the current state of a server in the FSM, the framework is able to determine the actions to perform. By limiting the amount of state required to make progress, the framework is able to easily recover fault.

Automated Incident Management System (AIMS) was deployed in a cloud, and during the deployment a few interesting additions were made to the framework: (1) to ensure that bugs in the system do not take down the entire datacenter, a circuit breaker was added to limit the number of elements to take action on; (2) to ensure that the system remains quiet during a planned outage, they introduced the ability of a system freeze for some fraction of time; (3) the authors included several filtering rules to reduce false positives; (4) they included self-healing properties to allow the system to write temporary logs that can be used by the framework at a future date to support future actions; and (5) due to the asynchronous nature and lag between diagnosis and corrective action, work-flow validation was added to ensure that corrective action was still required before performing it.

Does the framework understand the relationship between different resources used by a cloud user to create a service? If so, is there a service for this level of information? No, the system is currently geared towards provider-side management. Is there a state machine for each virtual machine? Yes, there is one for each virtual machine. However, the system only needs to track the current state. If the current state of a server is tracked by the framework and can be determined by querying the server, which is the authoritative source of information? The server is the authoritative source of information, and before actions are performed on the server the system reevaluates the state of the server.

QoSaaS: Quality of Service as a Service

Ye Wang, Yale University; Cheng Huang, Jin Li, and Philip A. Chou, Microsoft Research; Y. Richard Yang, Yale University

VoIP is becoming an integral part of the enterprise network ecosystem. In this talk, Ye Wang provided a system for aggregating measurements from different locations. The system uses these measurements to estimate the performance of

different applications and to determine problems introduced by individual devices. Ye proposes that VoIP applications query the system to determine how to adapt to issues in the network and that operators can use the system to diagnose QoS degradation.

This work is prompted by problems encountered by users of the Microsoft Lync VoIP system, used by 7 million users. Debugging performance-related problems is difficult, as it requires information about how each system component affects the end-to-end quality, information that is currently missing.

To debug QoS issues in VoIP calls, the system collects data from different clients and network entities and uses an inference engine at a central location to determine problems. The system models the following entities: the network, the Microsoft media servers, and the VoIP clients. The system models communication between two VoIP clients as a set of media lines encapsulating the entities involved in the VoIP class. Inference is performed by running a maximum likelihood algorithm on entities in the media lines. When performing inference, information from prior calls is used; however, unlike other inference systems, the system takes into account session durations and weighs the information used in the inference according to length of the session. In running the inference engine on the data collected from the deployed Microsoft Lync system, Ye observed that most problems are networking problems and most entities do not drop packets.

Is this data already being reported by Microsoft? If so, what is this system adding? It is currently being collected by Microsoft agents, and the system adds aggregation and inference algorithms.

KnowOps: Towards an Embedded Knowledge Base for Network Management and Operations

Xu Chen and Yun Mao, AT&T Labs—Research; Z. Morley Mao, University of Michigan—Ann Arbor; Jacobus Van der Merwe, AT&T Labs—Research

Network management comprises all activities required to keep the network healthy while delivering a certain SLA. These activities consist of configuration management, fault management, and planned maintenance, among others. The network management systems required for each activity are created and used by very different teams. The domain knowledge used in creating these systems and the documentation for using these systems are all encoded manually in human-readable documents.

Xu argues that a machine-readable knowledge base is required to express the information from the different domains and ultimately provide easy access to this information. COOLAID provides a framework for capturing this knowledge as rules that can be run against a database containing information on the network systems. The rules and abstractions made by COOLAID are machine readable. Working backwards from COOLAID, with the rules stating how different information interacts, the system can extract event correlation and determine which items to monitor and how frequently to monitor them. Furthermore, these rules can prove useful in determining the holistic impact of automated procedures and can be used to determine how and when to schedule tasks.

Can they automate all management tasks? It may not be profitable to automate all tasks, and some tasks may require some manual interaction due to diversity. Furthermore, guard rules may be required to ensure that the amount of evil is limited. Although the system requires experts to determine rules currently, can they create rules from the data itself? While it may be possible to mine the rules from the data, such an approach assumes that all data is correct and that data is comprehensive; however, there may be certain rules that will be absent from the data. Can the vendors, ISPs, and operators work together to determine how to standardize and express information? Yes, they can, and KnowOps is a step towards such standardization.

Network Management

Summarized by David Shue (dshue@cs.princeton.edu)

Using Hierarchical Change Mining to Manage Network Security Policy Evolution

Gabriel A. Weaver, Nick Foti, Sergey Bratus, Dan Rockmore, and Sean W. Smith, Dartmouth College

Security goals are often expressed in policies that ultimately affect system behavior. If you don't change the policy, the system becomes increasingly vulnerable to exploits. If you do change policy, implementation and change tracking complexity can explode. To combat the potentially unbounded complexity induced by evolving polices, Gabriel Weaver presented a mechanism to detect and determine the changes in hierarchically structured documents to track evolution and change, and ultimately help policy makers make better decisions.

Existing techniques such as changelogs (e.g., Adobe redlining, Word track changes) are insufficient for pinpointing the scope and extent of changes. By parsing the document into a hierarchy, as per a language grammar, Gabe's method can easily distinguish between large and small changes. At the core of the technique are two distance metrics used to compare document trees: Tree Edit Distance (differences

in subtree structure) and Word Edit Distance (differences within leaf text blocks). In comparing the output of the technique to a manually documented changelog, Gabriel found that 9 out of 178 reported changes were never made in the revised document, demonstrating the utility of the approach for edit verification.

The general technique can be applied to any well-structured document type, including router configuration (e.g., Cisco IOS), to help analyze and track the gap between security policy and implementation evolution. Current techniques based on RANCID do not leverage the structure of IOS and spot textual differences that are semantically equivalent. Change mining, on the other hand, analyzes the logical structure, filtering out order-invariant changes. Moreover, by constructing a fully diffed parse tree, the technique enables multi-level change querying based on tree prefixes, i.e., /root/interface for all interface-related changes. A user can drill down further to explore structural differences at finer granularity: interface block -> line card type -> specific interface.

Anees Shaikh asked whether the technique could rank changes based on impact or importance. Gabe said that is future work. Kobus Van der Merwe asked what the user actually queries in the system. The system parses the documents and the resultant paths are then loaded into a DB for querying. Ye Wang asked whether change mining is limited to just security policy. The technique is applicable to any well-structured text that exhibits hierarchical structure. Morley Mao asked whether the technique can detect invariance. Yes, by modeling the parse tree, which can filter out irrelevant textual changes.

Towards Automated Identification of Security Zone Classification in Enterprise Networks

HariGovind V. Ramasamy, IBM Research; Cheng-Lin Tsao, Georgia Tech; Birgit Pfitzmann and Nikolai Joukov, IBM Research; James W. Murray, IBM

Enterprise networks are partitioned into multiple zones of criticality where devices of similar security requirement are placed into the same zone (e.g., Internet, intranet, DMZ with firewalls between, etc.). Although the number of classifications (colors) may be small, there are often a large number of zones per color created to support diverse organizational and application isolation needs. Unfortunately, enterprise network administrators must manually track the zones in ad hoc documentation, which can be tedious and error-prone. To combat this problem Hari introduced BlueGates, a system that examines application-specific, flow-level access control to facilitate automatic security zone classification.

BlueGates relies on a defined security policy matrix that prescribes the allowed traffic between security zones—e.g., only port 80, ssh, all traffic, authentication required, etc.—as the canonical metric for classification. By gathering the actual flows allowed in the network using existing monitoring tools and comparing them against flows permitted by policy, the system can then infer zone colors. With the policy matrix, the set of hosts to analyze, and a small set of known classifications as input, the system generates a full delineation of network zones, their colors, and the interconnections between zones. Indeterminate hosts are assigned all colors. If all colors are infeasible for a host, then the system flags a potential conflict in policy and configuration. Future work on BlueGates includes loosening the assumption that the network configuration fully complies with security policy and evaluating the system on a large-scale infrastructure.

Kobus Van der Merwe noted that the system assumes that the security policy exists and can be properly represented as input. Hari said that this is true, someone must glean the information and perform data entry, but polices change on an infrequent basis compared to network configuration. Kobus followed up by asking how they know that the model is correct. Verification of the model and output with document authors and operators is an iterative process. What if configuration does not comply with the specification? While the solution may be imperfect given the assumptions, any useful though potentially flawed tool is helpful. Would reachability analysis between networks arrive at the same conclusions but in a simpler fashion? Simple flow-level reachability ignores per-application (protocol port) ACL and security requirements. Theo Benson asked whether switch configurations could provide similar information without the need to intrusively probe the hosts. Accessing switches and firewalls often raises more alarms than probing the hosts themselves.

Simplifying Manageability, Scalability and Host Mobility in Large-Scale Enterprise Networks using VEIL-click

Sourabh Jain and Zhi-Li Zhang, University of Minnesota—Twin Cities

With the increasing degree of mobility (mobile devices, VM migration) comes a new host of networking challenges: scalability, mobility, availability, manageability, etc. Existing layer 2 (L2) techniques support plug and play but at the expense of flat routing and the reliance on broadcast protocols. Layer 3 (L3) routing is scalable but not amenable to mobility and still requires control-plane flooding. Moreover, simple address changes require significant modification to network state (firewalls, DHCP servers, etc.). To surmount these challenges, Sourabh presented VEIL-click, an L2 (Eth-

ernet) network architecture with built-in support for scalable mobility and fast-failure rerouting.

In VEIL, IP addresses become persistent unique identifiers that remain immutable across network changes. To support topology-aware location-based routing, VEIL assigns each NIC a Virtual ID (L2 MAC) that encodes the network location. Routing on the VID is transparent to end hosts and compatible with existing Ethernet switches. On startup, VEIL-enabled switches in the network communicate with a centralized controller that assigns topology-dependent VIDs to each switch. Each VEIL switch then acts as an access gateway, assigning VIDs to each host (32-bit switch VID and 16-bit hostID) and mapping persistent IPs to VIDs by intercepting ARP requests.

Together, the VEIL switches form a DHT to distribute the IP-to-VID registry and forward traffic based on VID. When an ARP request arrives at a VEIL switch, it is resolved via unicast to the appropriate DHT root. A translation rule at the ingress VEIL switch maps the source MAC to VID for return traffic, and a corresponding rule at the egress VEIL switch maps the VID to actual MAC for final delivery to the destination NIC. To support mobility, when a host moves to a different point in the network the access VEIL switch assigns a new VID and updates the IP-VID mapping at the corresponding DHT node. An update instructs the VEIL switch responsible for the old VID to forward packets to the new VID mapping. Sourabh has built a VEIL prototype based on the Click modular router and showed a graph demonstrating a 1–2 second TCP flow disruption for mobility between wireless networks.

Morley Mao asked what the impact would be for real applications (VoIP) and not just at the flow level. Sourabh said that this is future work. Ye Wang asked about scalability for a large number of clients and switches. One could create a hierarchy of VEIL networks based on L3 routing. What is the performance impact of modifying Ethernet headers? The overhead is less than TRILL. Chang Kim asked why not rewrite IP addresses. L2 name resolution already exists (ARP) and the different sizes of IPv4/v6 would also cause issues for VID-based routing. What is the increased risk of address spoofing due to the extra indirection? Mitigating the attack would require a means of verifying the owner of the IP address, such as a PKI-based certificate mechanism.

Mini-panel

Morley Mao asked if there is anything at odds between security and performance, or are there any improvements vendors can make to facilitate management?

Hari Ramasamy said it would be useful to have vendor-provided tools for constructing an end-to-end view of the network. Someone in the audience remarked that the ConfigChecker research project does just this, providing a global end-to-end perspective of the network. Hari also noted that the teams looking at performance and security are different and often have different goals that may conflict.

Gabriel Weaver posited that finding out the issues that vendors have can inform and direct new research work.

Someone asked a VEIL-specific question: Can there be multiple VIDs for an IP address? Sourabh Jain answered that there is one IP per NIC and only one VID per IP.

Someone wondered what happens for multi-homed devices and the scalability to determine liveness. This resulted in a discussion that can be summarized as "liveness mechanism doesn't scale (n^2)."

Morley Mao had a question to the ISPs: What are the insights into problems from homegrown solutions—that is, what are the underlying principles you've gleaned? Kobus Van der Merwe answered that you need to think about the effect on services, not just the lower layers of the network stack.

Morley Mao asked about the challenges in terms of validation with real data. Hari Ramasamy replied that firewall configs are not taken from the production firewall but rely on a snapshot instead, which could lead to stale analysis. These are inherent limitations of the internal processes. Vendor tools could help by anonymizing configs. Someone else wondered if trouble tickets could be used as a source of information. Hari replied that they could be a useful source of info.

Gabriel Weaver asked about analyzing security changelogs. Hari said that this requires a loop between researcher and practitioner to obtain data and verify results and to determine the usefulness of the results.

Someone asked Gabriel Weaver whether you can have a subtree exist and assign a metric to state the level of interest/non-interest. Gabriel replied that you can determine what to evaluate/ignore. The technique can parse human-readable and machine-readable specifications: RFCs, configuration, etc.

Datacenter Networking

Summarized by David Shue (dshue@cs.princeton.edu)

Enabling Flow-level Latency Measurements across Routers in Datacenters

Parmjeet Singh, Myungjin Lee, Sagar Kumar, and Ramana Rao Kompella, Purdue University

Many of the online services populating datacenters today provide latency-sensitive customer-facing applications.

Troubleshooting latency anomalies at the end host requires simple local measurements, but flow-level latency measurements in the network fabric have proven difficult. Previous work on Reference Latency Interpolation (RLI) can perform fine-grained measurements but requires that every router be RLI-enabled. In this work, Myungjin Lee proposed a lighter-weight approach to router-level flow-latency measurement: RLI across Routers, or RLIR, which only requires every other router to support RLI by trading off latency localization granularity (from link to path segment) for cost and ease of deployment.

In the RLI architecture, router pairs send reference probe packets with embedded timestamps between an ingress interface on one router and an egress interface on the other to determine link latency. By exploiting delay locality, where closely spaced packets experience similar queueing delay, the routers can linearly interpolate between reference packet latencies to estimate nearby regular packet latencies. In a partial deployment scenario, ECMP may cause packets to take different routes between routers, violating the delay locality principle for the measured paths. Cross-traffic along the path also complicates the reference packet sampling rate estimation.

To address these issues, RLIR senders associate with all intermediate RLIR routers along its flow paths, and RLIR receivers examine the source IP prefix to distinguish reference from regular packets for proper flow latency correlation in the upstream direction (top-of-rack to core router). Additionally, an RLIR receiver may also need to reverse compute the ECMP splitting or rely on packet marking to determine the full flow path in the downstream direction (core to top-of-rack). For sampling rate estimation, RLIR assumes worst case utilization along the path segment and rate-limits reference packets accordingly. Evaluated in simulation using an OC192 link trace, RLIR was able to achieve less than 10% relative error in mean estimates at 93% utilization for 70% of the flows and similar relative errors in standard deviation estimates for 90% of the flows. For bursty traffic, RLIR was able to capture the differences in mean latencies to aid in identifying and isolating adversarial cross-traffic.

Jeff Mogul wondered how far you can push deployment sparseness. Myungjin said it is a natural tradeoff between cost and localization granularity. Chang Kim asked why the technique requires hardware assistance. Myungjin replied that the reference packet injection with router timestamps and linear interpolation estimates at line speed require fast-path hardware support.

OpenFlow-Based Server Load Balancing Gone Wild

Richard Wang, Dana Butnariu, and Jennifer Rexford, Princeton University

Effective load balancing for datacenter-based replicated services should handle aggregate request volume, scale out across available replicas, and provide a programmable interface, while remaining cost-effective. Existing software load-balancers are flexible, but are throughput limited. Dedicated hardware load balancers lose programmability and can be complex and expensive. Enter OpenFlow, which combines a simple distributed hardware data plane with a flexible centralized software control platform. In this talk, Richard Wang introduced a scalable and flexible load-balancing solution based on OpenFlow wildcard rules.

Although OpenFlow supports a large number of exact match rules, rule-per-flow load-balancing is still bottlenecked by the microflow rule table size and controller throughput, since the controller must act on the first packet of every flow request. On the other hand, although smaller in number, OpenFlow wildcard rules can match on src IP prefixes to bulk distribute incoming flows across server replicas without any additional controller interaction. Assuming a uniform distribution of requests across the client (source) IP address space, wildcard rules can be used to split flows across replicas according to a (power of two) weighted distribution. Additional rule savings can be achieved by combining rules with a common source IP prefix to achieve a particular weighting for a given replica.

Since wildcard rules direct flows to replicas in aggregate, the controller must intercede whenever the wildcard rule set changes, to preserve flow affinity. When a new wildcard rule overlaps with an existing rule, the controller enters a transition period before installing the new rule. During this interval, the old rule directs packets to the controller, which then installs microflow rules to direct existing flow packets (non-SYN) to their original replica, while new flows (SYN) are directed to the new replica. After the transition period expires, the controller installs the new wildcard rule. Any existing microflow rules will expire once the flows terminate, due to the OpenFlow rule idle timeout. Future work includes supporting a broader range of client IP distributions and scaling the transition period reliance on microflow rule computation with multiple controllers.

Someone asked how the duration of the transition period is determined and how the controller handles long-lived flows. Richard answered that the default is 60 seconds, which is a tradeoff between flow affinity and system responsiveness. What is the scale difference between wildcard and exact match rules tables? Richard said 36k exact match vs. 100 wildcard (TCAM-based). Why can't the wildcard rules

match on the least significant IP bits for better entropy? Currently, OpenFlow only supports prefix wildcards. Would the OpenFlow 1.1 spec which supports hash-based rule anycast obviate the need for wildcard matching? The ensuing discussion concluded that it may be a while before hardware vendors support the 1.1 spec, and consistent hashing based on the typical flow 5-tuple could be used to distribute flows across the replica set.

Kobus Van der Merwe wondered how much flexibility is truly necessary, given that the set of functionality provided seemed standard. Richard replied that controller programmability enables feature evolution, though a more detailed feature set and performance comparison to F5 and Zeus load-balancers that are available as VMs for free evaluation may be useful.

Online Measurement of Large Traffic Aggregates on Commodity Switches

Lavanya Jose, Minlan Yu, and Jennifer Rexford, Princeton University

Monitoring and identifying large traffic aggregates improves network visibility and fosters a deeper understanding of the latent traffic structure: that is, heavy users, popular Web sets, anomalous DDoS traffic patterns, etc. In this work, Lavanya Jose presented an OpenFlow-based approach for online aggregate traffic measurement, with a particular focus on identifying hierarchical heavy-hitter traffic, using a cheaper and simpler commodity platform compared to previous approaches.

Coarse-grained aggregate views may overlook individual culprits, while fine-grained monitoring could miss the heavy subnet forest for the mid-weight IP trees. Hierarchical heavy hitters (HHH) use a multi-resolution approach to identify both individual and aggregate traffic sets of interest. HHH forms a monitoring trie (prefix tree) based on subnet prefixes. In this prefix tree, any leaf node that consumes more than a fraction T of its link capacity is marked as a heavy hitter. To filter redundant information, parent nodes are only considered as heavy hitters if the total link utilization of non-HHH children still exceeds T. Prior work on HHH (Autofocus), which analyzed offline packet traces, was accurate but slow. Building custom hardware for online monitoring would be prohibitively expensive. Instead, by leveraging OpenFlow's programmable data plane monitoring facilities with wildcard rules provides a more feasible monitoring solution.

OpenFlow wildcard rules make a good match with HHH, despite the limited pool of TCAM-powered rules. Since OpenFlow rules respect a strict priority ordering, high and low resolution rules can form an overlapping monitoring hierarchy based on set-differences. The controller uses at most 2/T rules to monitor and detect heavy hitters by iteratively adjusting wildcard rules over each monitoring interval M. If a prefix monitoring rule detects that a node exceeds T, the controller adds additional rules to drill down and monitor the children. Otherwise, if the utilization drops below T, the node monitoring rules are dropped. Any leftover rules are used to monitor HHH parent nodes that are close to the T threshold. In a simulated evaluation on a packet trace from CAIDA (400K packets/s) with a monitoring interval spanning 1–60 seconds, the framework was able to identify 88–94% of the T = 10% HHH with only 20 rules.

Someone asked about the complexity of the algorithm. Lavanya replied that the technique is linear in the number of rules used, which corresponds to the number of HHHs (2/T), not the size of the data. How does the algorithm bootstrap the monitoring? It starts with the largest (root) prefix first. What is the timescale of convergence? About 10–20 intervals.

Topology Switching for Datacenter Networks

Kevin C. Webb, Alex C. Snoeren, and Kenneth Yocum, University of California, San Diego

Large-scale datacenter networks house multiple tenants with a diverse range of applications. While most current techniques address network contention by adding capacity, applications may have differing and conflicting needs: MapReduce wants high bandwidth and MPI desires low latency. A simple MPI benchmark run with a background Hadoop job caused latency to increase 20% in the network. In this talk, Kevin Webb presented a system that constructs a virtual network topology according to application-specified properties: bandwidth, redundancy, latency, and isolation using the OpenFlow platform.

Applications submit routing tasks which describe the set of communicating end hosts, logical topology, and an allocation algorithm for the desired properties. The allocator specifies an objective function to evaluate prospective allocations, an allocation algorithm, and an annotation mechanism to mark and filter resources. Using these pieces, the allocator translates the application demands into allocations and routes in the network given the physical network view exported by a server-based topology. Once allocations are determined, the topology server annotates the network graph to mark the consumed resources and installs the routes into the network.

Kevin described three allocators implemented in the system: bandwidth driven, which finds the least loaded paths by building a max spanning tree over the remaining network capacity; resiliency, which searches the path space to find N disjoint paths between every host pair; and k-isolation, which builds a spanning tree with links used by, at most, k other

tasks. Simulations comparing the system to a full ECMP solution for resiliency and bandwidth and VLANs isolation show that the system can achieve resiliency and full isolation, where the ECMP/VLAN approach overachieved on resiliency but provided a smaller degree of isolation. Future work will focus on building the architecture and quantifying the degree of inter-application interference.

Kobus Van der Merwe inquired about the timescale of decision-making. Kevin said that the system responds to on-demand requests and makes decisions as quickly as it can. Chang Kim asked whether capacities were part of the application specification. Rate-limiting based on capacities would be helpful but is not yet included. Kobus asked why the system assumed a fixed set of servers and a malleable network when often the reality is the opposite for virtualized environments. The system should probably handle both VM placement and network resource allocation. Another questioner remarked that the allocation decisions may be good for isolation but not utilization, and multipath may work better. Kevin agreed that the bandwidth allocation is not optimal and will look into multipath in future work.

Mini panel

Someone asked if there is a standards body for defining relationships between multiple VMs, including their network requirements. Kobus Van der Merwe wondered if OpenFlow happens to be the best hammer available for implementation or is it the right switch abstraction? Richard Wang replied that OpenFlow simplifies control and implementation of custom switch behavior and allows substantial customization. Both Amazon and Google are starting to use it in real networks. Someone asked if the panelists had any thoughts on the Open Networking Foundation. An unidentified person responded that its primary goal is to harden and extend the OpenFlow spec for customers and vendors.