Dynamic Measurement-Aware Routing in Practice

Guanyao Huang and Chen-Nee Chuah, University of California Davis
Saqib Raza, Cisco Systems
Srini Seetharaman, Deutsche Telekom R&D Lab

Abstract

Traffic monitoring is a critical network operation for the purpose of traffic accounting, debugging or troubleshooting, forensics, and traffic engineering. Existing techniques for traffic monitoring, however, tend to be suboptimal due to poor choice of monitor location or constantly evolving monitoring objectives and traffic characteristics. One way to counteract these limitations is to use routing as a degree of freedom to enhance monitoring efficacy, which we refer to as measurement-aware routing. Traffic sub-populations can be routed (rerouted) on the fly to optimally leverage existing monitoring infrastructures. Implementing dynamic measurement-aware routing (DMR) in practice is riddled with challenges. Three major challenges are how to dynamically assess the importance of traffic flows; how to aggregate flows (and hence take a common action for them) in order to conserve routing table entries; and how to achieve traffic routing/rerouting in a manner that is least disruptive to normal network performance while maximizing the measurement utility. This article takes a closer look at these challenges and discusses how they manifest for different types of networks. Through an OpenFlow prototype, we show how DMR can be applied in enterprise networks. Using global iceberg detection and capture as a driving application, we demonstrate how our solutions successfully route suspected iceberg flows to a DPI box for further processing, while preserving balanced load distribution in the overall network.

Accurate and efficient network-wide traffic measurement presents multifaceted challenges. It involves not only configuring individual monitors for different measurement purposes (ranging from inferring flow statistics [1] to heavy-hitter identification [2], anomaly detection [3], and zooming in on traffic sub-populations [4]), but also deciding the placement of these monitors. All previous work in the latter domain defines some expected utility function for the monitored traffic and a cost function for the monitor deployment or operation. These functions are then used to identify the appropriate placement that delivers the most desired trade-off [5]. Such solutions typically assume long-term traffic characteristics and use static monitoring methods catering to specific traffic conditions. Hence, they are not capable of intelligently adapting to dynamic changes in traffic landscapes, network operating conditions, or measurement goals. It is quite possible for a flow of interest to avoid detection by not traversing the deployed monitoring boxes.

We addressed the limitations of such inflexible monitoring infrastructure in an earlier work called MeasuRouting [6], which proposes to assist traffic monitoring by intelligently routing traffic sub-populations over appropriate configured/deployed monitors. It uses routing as a degree of freedom to offset the disruptions caused by evolving measurement objectives or network/traffic characteristics. Figure 1 illustrates the basic concept of MeasuRouting. We assume that only router B is equipped with measurement capability. Suppose the network operator is interested in monitoring flow 2, which does not traverse B using default routing. To utilize the existing (fixed) monitors, MeasuRouting switches the route of flow 2 with another flow of similar size (flow 3), which enables the measurement of flow 2 while keeping the overall load balanced across the network. Reference [6] showcases the expected benefits of using routing to assist traffic measurements and how it it can be done without disrupting the network's traffic engineering (TE) performance. However, MeasuRouting only models a subset of the constraints a real-world deployment faces, and the benefits should therefore be interpreted as best case bounds for routing-assisted measurement. The theoretical framework in [6] deviates from practice in the following ways:

- It assumes a priori knowledge about presence of flows and their importance, which is hard to predict in practice.
- It assumes that any arbitrary subset of flows between an origin-destination (OD) pair can be aggregated to take a common forwarding action.
- It ignores practical routing constraints imposed by the routing substrate. It uses linear programming (LP) to solve the routing problem wherein each aggregate of flows may be arbitrarily divided across multiple paths as long as there are no loops.
- It works on independent snapshots of the traffic matrix without accounting for any flow arrival pattern, thereby

0890-8044/11/$25.00 © 2011 IEEE

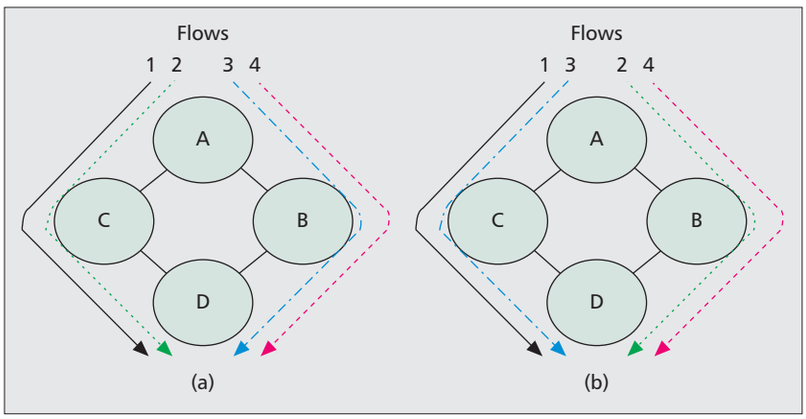

Figure 1. *MeasuRouting example: a) before MeasuRouting; b) after MeasuRouting.*

alleviating the need for rerouting.

In this article, we investigate how dynamic measurement-aware routing (DMR) can be realized in practice. DMR overcomes unpredictable changes in both traffic characteristics and monitoring objectives by dynamically routing important traffic sub-populations to some preconfigured monitors. We outline the challenges of DMR and build an OpenFlow [7] based prototype for enterprise networks, which is tuned toward a specific monitoring application: *global iceberg detection and capture*. We use counters of OpenFlow switches to assess initial flow importance and route potential iceberg traffic to a deep packet inspection (DPI) box, which is a typical example of any costly and complicated measurement/inspection task for important flows. We illustrate how iceberg flows can be gracefully routed with little or no influence on network performance. Our work serves as a first step to understand and achieve a functional DMR architecture.

The rest of the article is organized as follows. We discuss practical challenges for DMR in greater detail. We then present our system architecture and solution algorithms, introduce one example application, global iceberg detection and capture, describe experimental settings, and present evaluation results. We then conclude the article.

Challenges of DMR in Practice

In this section, we discuss three major challenges associated with realizing DMR in practice.

Dynamically inferring flow importance: The primary motivation behind routing-assisted measurement is to enable prompt reaction in dynamic environments. This involves changes in measurement objectives, network, or traffic characteristics, of which one cannot assume any priori knowledge. In DMR the inference of the flow importance (from a measurement perspective) is a dynamic process, wherein measurement tasks are continuously fine-tuned based on the most up-to-date information. Such a dynamic calculation/estimation of flow importance is a key prerequisite for making routing/rerouting decisions. Coarser-grained measurements, such as Simple Network Management Protocol (SNMP) statistics or uniform sampling [8], may be used to configure DMR to direct traffic toward more sophisticated measurement instrumentation (e.g., devices used for data streaming [9] or DPI). The challenge faced by DMR is how to dynamically learn flow importance at distributed points and communicate it back to the routing control plane.

Aggregating flows into flowsets: Routing and forwarding table entries need to be conserved for a number of reasons. For instance, only limited tertiary content addressable memory (TCAM) or fast-memory resources are available to house the routing table. Larger routing tables may also inflate the lookup latency. In practice, the number of routing table entries is conserved by aggregating flows into flowsets. We define "flowset" as any aggregation of flows as long as they share the same forwarding decisions. We refer to the problem of aggregating flows to flowsets as the flowset configuration problem. One consequence of flowset aggregation is that flows of the same flowset must take the same action (e.g., be forwarded across the same port[s]). Aggregation, therefore, restricts our ability to make fine-grained monitoring-aware routing decisions. MeasuRouting [6] assumes that arbitrary sets of flows between the same OD pair can be aggregated into a flowset. It also assumes that the number of flows for each flowset follows a certain distribution. However, in reality, flowset aggregation is highly dependent on the underlying routing substrate. For instance, in longest-prefix-based IP forwarding a flowset is characterized by a destination IP prefix. DMR in practice must be cognizant of and obey all flow aggregation constraints of the underlying substrate.

Dynamically routing and rerouting flowsets: The configured flowsets may need to be dynamically routed based on the learned or estimated flow importance. The objective is to maximize some utility function of measured flows while minimizing any cost incurred by deviating from the default routes. These costs include additional path stretch and maximum link utilization. In a dynamic setting, the importance of an existing flowset might change when its flows change properties or new flows arrive. The resource occupied by an initially important flowset may need to be allocated to a new or re-constructed flowset. The cost, therefore, also includes the number and impact of routing table changes required to perform the rerouting. DMR in practice must solve the dynamic routing problem in a timely fashion, minimizing disruption to network performance whilst obeying flowset configuration constraints imposed by the routing substrate.

The above challenges generalize to all kinds of underlying routing substrates and networks. Addressing these challenges link traffic engineering and monitoring together, making routing-assisted monitoring feasible in a dynamic environment.

DMR Proof of Concept

To study the practical implementation issues of DMR, we present an OpenFlow-based prototype that addresses the three challenges. The prototype is designed for enterprise or data center networks. The motivation behind doing so is twofold. First, we consider an enterprise network to be the ideal candidate for DMR. Administrators may have highly customized and rapidly evolving measurement objectives in contrast to more stable measurement requirements in backbone networks. Furthermore, the traffic rates are low enough to attempt DMR without the high stakes involved in backbone networks. Second, OpenFlow [7] has been shown to be a practical control mechanism for enterprise or data center networks. The ability to dynamically program forwarding behavior of network switches is an important feature that is leveraged by our prototype.

OpenFlow Architecture

OpenFlow is an open standard that decouples control from switches in such a way that a remote controller can dynamically change flow table entries on OpenFlow-enabled switches. Each flow table entry has matching rules described by the fol-

lowing 11 tuples in the packet:

$$\underbrace{<srcmac, dstmac, vlan_{id}, vlan_{priority}, type,>}_{\text{Data Link Layer}}$$
$$\underbrace{<srcip, dstip, proto, tos,>}_{\text{Network Layer}}\underbrace{<srcport, dstport,>}_{\text{Transport Layer}}$$

With OpenFlow architecture, a packet gets forwarded if it matches a flow-table entry. Whenever a new flow has no preconfigured entry, the first packet will be forwarded to the controller, which in turn calculates the route and populates corresponding flow-table entries. A flow-table entry can be specified by the above 11 tuples with an additional field denoting the incoming port of the packet. Certain fields can be wildcarded (i.e., unspecified) to aggregate flows into flowset. The controller can also query the switches for an update of flowset statistics. When there is no packet matching an entry for a certain period, the entry automatically gets deleted and a flow-expire message containing statistics of the expired entry will be sent to the controller. The statistics collected from flowquery and flow-expire messages can be directly used by the controller to control flowset routing.

We now discuss how each of the challenges enumerated earlier is addressed in our prototype solution.

Learning Flow Importance

For many applications, flow importance is estimated by the size of flows. Example applications include detection of elephant flows and traffic churns, and estimation of flow size distribution [9]. In OpenFlow, collecting flow sizes is easy if we have an unconstrained number of flow table entries, since one flow table entry corresponds to a single flow. The controller can retrieve the byte/packet counts for each flow using flowquery/flow-expire messages in the OpenFlow protocol. However, in practice, multiple flows are grouped into a flowset, and the flowset corresponds to a single flow table entry. Consequently, byte/packet counts are aggregated over many flows. To overcome this, our prototype periodically disaggregates a small portion of flowset entries into entries of individual flows, based on flowset statistics. It then maintains/queries individual flow statistics for a certain period to estimate their importance. The procedure is repeated such that every flow gets its importance estimated. In DMR, the estimated important flows will be routed to more complicated monitors for better measurement. It tolerates estimation errors in assessing flow importance.

Flowset Configuration

As mentioned earlier, flow table rules can be exactly specified by the 12-tuples or wildcarded to aggregate flows into flowsets. The rules are described using datawords in a TCAM. In order to map a set of flows into the same flow-table entry, we require that a TCAM dataword be wildcarded on certain tuples. TCAMs allow all or a subset of the bits in the data search word to have a "don't care" state (i.e., any value for that bit will count as a match). This allows a single TCAM dataword to possibly match multiple flows. The current OpenFlow release imposes restrictions on how the "don't care" bits are set. All the fields except *mac* and *ip* addresses may be constrained to have either all their bits specified as "don't care" or none at all. The *mac* and *ip* fields can specify a postfix of its bit string to be "don't care." These constraints restrict the flexibility to configure flowsets, but must be adhered to while deploying a practical solution.

Furthermore, our enterprise networks has flat layer 2 addressing as opposed to hierarchical layer 3 addressing. This limits our ability to aggregate flows that are destined for a common egress switch. In order to solve this problem, we leverage on the solution in [10], which proposes to use a hierarchical pseudo medium access control (PMAC) address to encode a host's position in the topology. All hosts behind a switch will have the same prefix in their assigned PMAC addresses. The hosts remain oblivious to the assigned PMAC addresses, with their associated switches performing the appropriate PMAC to MAC header rewriting. The solution allows OpenFlow to group together flows destined to the same egress switch into the same flowset.

As mentioned earlier, there is a budgeted number ($b = m + n$) of flowsets (TCAM entries per switch) for each OD pair. We allocate n of these flowsets for default routing and learn individual flow importance. We assign the m most important flows to the m flowsets such that they can be routed differently from common flows. This is achieved by setting the fields of the TCAM dataword equal to the 12 tuples. All common flows are mapped to one of the n flowsets, which are constructed by wildcarding certain fields of the TCAM dataword. In OpenFlow, TCAM entries with exact matches always take the highest priority; the m important flows will not be forwarded by any of the n flowset rules.

Flowset Routing

In our proof-of-concept DMR implementation, we investigate one simple heuristic solution for the flowset routing problem. It is an extension of 1/R routing [11]. In 1/R routing, link weights are assigned to be the reciprocal of residual link bandwidth, and then the shortest paths are computed for all node pairs. In OpenFlow, the controller can periodically query each link's utilization. We assume 1/R routing to be the default case, which is reasonable if the operator intends to distribute the traffic evenly across the network. The paths calculated in this way are referred to as *1/R paths*. We devise concatenated 1/R routing (CR) to route important flowsets, as explained in the following. The definition of "important" is specific to different measurement applications.

CR is designed for the case where there is only one monitor in the network. It chooses the path with the smaller cost (in terms of sum of link weights) from the following two candidate paths. The first candidate path is a concatenation of a subpath from the *src* switch to the monitor, and the other is a subpath from the monitor to the *dst* switch. Both subpaths are calculated by 1/R routing, and loops are avoided by iteratively excluding already selected switches. The second candidate path is constructed in a similar manner, with a reversed order of calculating the two subpaths.

In the case where there are multiple monitors, we propose another heuristic solution called weighted 1/R routing (WR). WR assigns link weight as the reciprocal of a monitor weight multiplied by residual link bandwidth. The monitor weight is large for links attached to the monitor and small for all other links. The smaller link weight for the links attached to the monitor will attract traffic toward it. These modified link weights are used to compute the shortest path for important flowsets. Our initial results show the the performance of CR and WR are comparable for a single-monitor case. For simplicity and illustration purposes, we only present CR in our evaluation.

In DMR, flowsets might get rerouted once their importance and network conditions are dynamically updated. Therefore, we consider the following two variations for CR:

- R, reroute existing flowsets if their importance changes.
- NR, do not reroute existing flowsets even if some of them become important, but only route a new flowset in a moni-

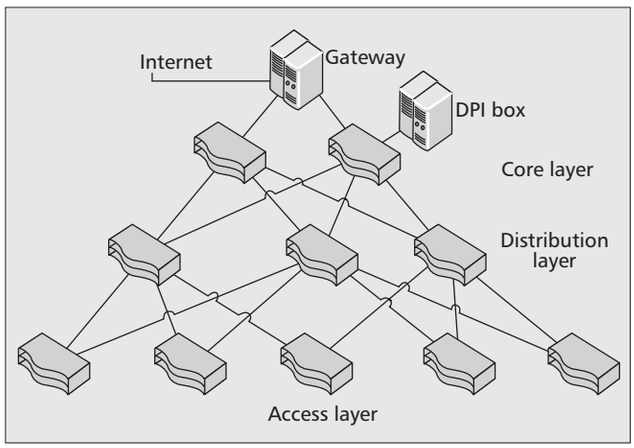

Figure 2. *Experiment topology.*

tor-aware manner.

In order to avoid congestion, we amortize the redirection tasks over time. In our experiment, we find that there is no need to explicitly reroute unimportant flows for load balancing. Amortizing redirection tasks is sufficient.

To summarize, our DMR prototype implemented in OpenFlow conducts the following procedures:

- It iteratively splits flowset entries into small entries for individual flows. It learns, estimates, and updates the importance of flows by querying switches or utilizing flow-expire messages. Important flows are routed using a separate flow table entry while the rest are aggregated.
- It routes new flows as they arrive based on known flow/flowset information. This includes deciding whether/how to group the flow into a flowset and how to route the flowset. It reroutes/reconfigures existing flowsets based on dynamic traffic conditions.

DMR Case Study: Capturing Global Icebergs

To showcase the effectiveness of our OpenFlow-based DMR prototype, we consider one driving measurement application: *global iceberg detection and capture*. A global iceberg is defined to be an "item" whose aggregated count value is above a certain threshold. In this article, we define the "item" to be all flows destined to the same *nw_dst* (IP destination address), that is, an *nw_dst* that attracts a lot of traffic (larger than μ fraction of total amount) from distributed hosts. Such icebergs are of interest for different network management tasks, such as detecting a distributed denial of service attack or identifying botnet command and control traffic. Detailed security analysis (e.g., DPI) allows postmortem analysis of events seen in the network and payload properties of the iceberg traffic in order to determine its potential source. However, capturing payload is often an expensive process that requires dedicated hardware or vast storage capacity for captured traces. As a result, operators can rarely afford to deploy DPI agents at every single node. Instead, a handful of DPI boxes are placed at strategic locations with limited storage resources. Hence, it is desirable to route flows destined to iceberg *nw_dst* across these DPI boxes installed at selected locations.

In our case study, we deploy DMR to learn the existence of a global iceberg and reroute the flows belonging to the iceberg toward the DPI box. To assess the *quality of the traces* captured for subsequent DPI, we define flow utility I_x to be the proportion of the size of its *nw_dst* and μ of total traffic amount if the *nw_dst* is the iceberg; otherwise, it is 0. I_x is calculated offline to present the ground truth. We use b to denote the measurement utility gained by capturing iceberg flows,

$$\beta = \sum_x I_x \varepsilon_x^*,$$

where ε_x^* is the packet number for flow x that is forwarded to the DPI box. We define the measurement improvement as a ratio, $\Delta_\beta = (\beta_{DMR} - \beta_{default})/\beta_{default}$, where β_{DMR} is the measurement utility gained with DMR, and $\beta_{default}$ is the utility achieved by default routing.

Testbed Setup and Performance Metrics

The current release of OpenFlow is more suitable for handling enterprise network due to its limited number of TCAM entries (around 2000) used to create the flow table. To test DMR on OpenFlow, we would ideally have complete layer 2 topologies of enterprises and traces on every link. Unfortunately, such information is not accessible due to privacy concerns. Instead, we build a topology as shown in Fig. 2. The topology is typical for a tree-like enterprise network where the central gateway is at the top layer. We assume there is only one DPI box (monitor) in the network. We use packet level traces from the Lawrence Berkeley National Laboratory campus network [12]. We replay one trace to each switch with moderate rate (around 60 packets/s from each switch) to imitate a real OpenFlow-enabled campus network. The small rate is due to current limitations on the number of entries (around 1500) per OpenFlow switch. Our prototype can support larger data rates with an increased entry number supported by the switch. We also scale link bandwidth such that the maximum link utilization (MLU) is around 20 percent.

In our experiment, we estimate iceberg identities on the fly based on most recent measurement; for example, an *nw_dst* is *suspected* to be an iceberg if its size is larger than μ fraction of *current* traffic volume. We vary iceberg threshold from μ = 1 percent → 5 percent. Once a flow or flowset is estimated to be important (i.e., part of an iceberg in this case), we apply DMR to route it toward the DPI box. We describe the various flowset configuration and routing scenarios in the next section.

Besides Δ_β, we are also interested in the following cost metrics:

- Δ_{MLU}: We consider maximum link utilization (MLU) as our traffic engineering (TE) metric and define Δ_{MLU} as $MLU_{DMR} - MLU_{default}$; that is, the difference between MLU when DMR is deployed and MLU when default routing is used.
- κ: The number of rule changes needed to perform rerouting.
- L: Average hop count for all the flows.

Other performance metrics include average packet delay and throughput. In OpenFlow, it takes around 10 ms to create a flow table entry per flow. Such delay is trivial because only the first packet will trigger the entry write. Since there is no packet loss, the packet delay is closely related to L, and the overall throughput stays the same.

Experiment Scenarios

Since we rely on OpenFlow counters (associated with flowset entries) to estimate the size of flowsets, the choice of flowset configuration has an impact on how well DMR assesses the importance of flows. To benchmark the performance of DMR, we consider the following three experiment scenarios.

(Case 1) Flow routing: In flow routing, each entry in an OpenFlow switch is defined for an exact match of 12 tuples;

that is, each flow has its own flow table (flowset) entry regardless of its importance. Each flow has its estimated utility I_x^* calculated in the same way as I_x, based on current size of *nw_dst* and traffic volume. Based on I_x^*, CR directly forwards estimated iceberg flows to the DPI box. Flow routing is the finest granularity for flowset configuration, thereby providing the prototype with greatest control and highest expectation of monitoring improvement. The downside of this approach is that it requires a large number of flow table entries, and hence may not be feasible in practice.

(Case 2) Static flowset routing: In this approach, flows are grouped into flowsets by wildcarding certain *fixed* tuples. In our setting, the flow table entries are matched to 6 tuples: *dl_type*, *dl_vlan*, *dl_src*, *dl_dst*, in port and the wildcarded *nw_dst*. We ignore following fields of the 6 tuples: *nw_proto*, *nw_src*, *tp_src*, *tp_dst*, *nw_tos*, *dl_vlan_pcp*. We assign the same PMAC [8] *dl_src*/*dl_dst* address to all the flows initiated from/to the same switch. We do not ignore *nw_dst* since it is used to define an iceberg. However, we wildcard the last 26 bits of it to save entry numbers. In reality, *dl_dst* itself is sufficient to forward packets to the destination switch. However, such high-level aggregation prevents us from learning more detailed flow information. Our wildcarding saves 66 percent of entries compared to flow routing. Due to aggregation, individual flow sizes are no longer visible. Since the OpenFlow counter is associated with flowset entries, the estimated size is for aggregated flows that share the same entry.

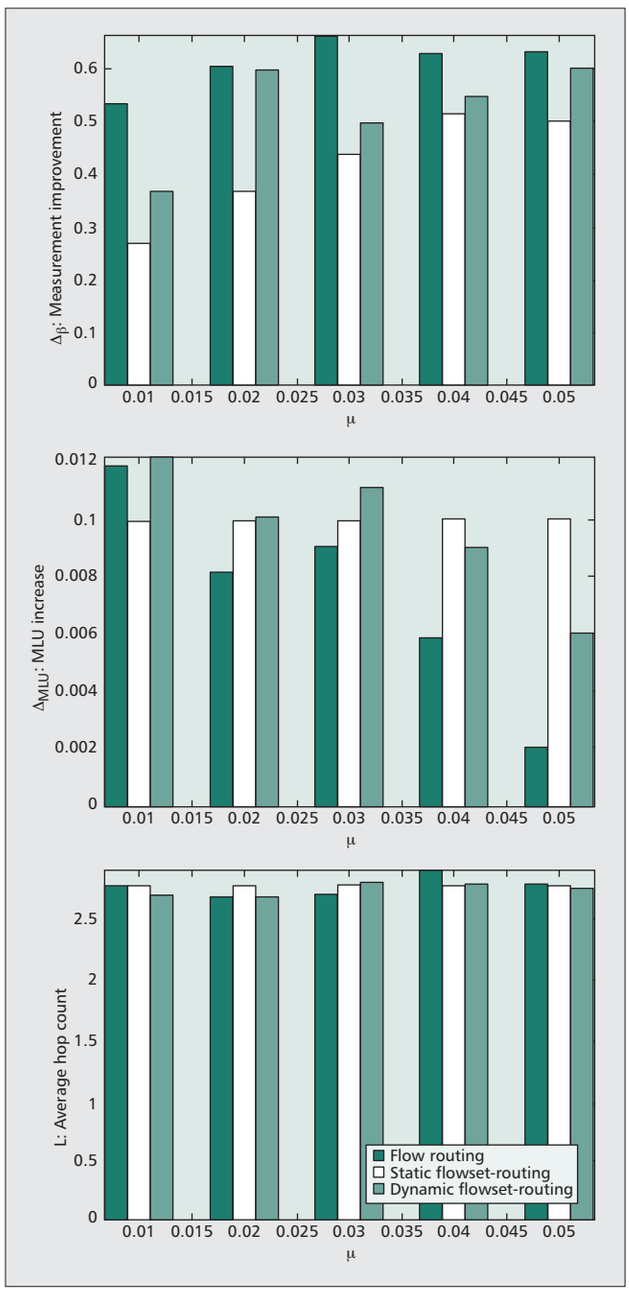

Figure 3. *Advantage of dynamic FlowSet-Routing.*

To calculate flowset utility we define *nw_dst** to be the wildcarded *nw_dst*. There are therefore 64 (2^{32-26}) distinct *nw_dst**. A flowset is important if its *nw_dst** is among the most popular (i.e., its size is larger than μ* of total count). Similar to I_x^*, we calculate the estimated utility for *nw_dst** and perform CR accordingly.

Note that there is a mismatch between μ* and μ since μ* is defined for wildcarded flowsets; that is, because an *nw_dst* is important does not necessarily mean its *nw_dst** is also big. It is therefore difficult to decide a proper μ*. We test the performance on different μ*, but only present results for μ* = 5 percent and μ = 1 percent → 5 percent earlier.

(Case 3) Dynamic flowset routing: Here, flows are grouped into flowsets in the same way as in static flowset routing, except that the module now periodically splits some of the flowset entries to query statistics of individual flows (as described earlier). The sizes of individual *nw_dst* then become visible. A flow will be routed separately once it is estimated to be an iceberg (hence important), while common flows can be aggregated again to share one wildcarded entry and perform 1/R routing. Note that we only need to split the flowset at one switch to learn flow sizes, and keep the same flowset configurations at other switches.

Dynamic flowset routing maintains three separate identities: OF (ordinary flowsets), CF (candidate flowsets), and IF (important flows). CF are the flowsets that divide their entries at one switch to learn individual flow sizes. For each period (20 s in our experiment), the module performs three tasks. First, it reclassifies some flows from CF to IF if the rate of *nw_dst* for the flow is larger than μ of overall traffic rate. Second, it moves the remaining CF back into OF. Common flows of previous CF will again be aggregated by wildcarded flowsets. Lastly, it selects some *nw_dst** and their corresponding OF into CF. In our program we pick one random *nw_dst** and the largest (more than μ* of traffic amount) *nw_dst**s, and partition their corresponding flowsets into CF. It then splits CF's entries to access individual flow importance. Every wildcarded flowset uses 1/R routing, while important flows are routed toward the DPI box with CR.

Evaluation Results

We now present evaluation results for the three scenarios using CR routing with NR and R variations.

Advantage of Dynamic Flowset Configuration — We first compare the performance of dynamic flowset routing against flow routing and static flowset routing in Fig. 3, using the CR/NR routing algorithm. Since NR derivation does not redirect traffic, κ remains 0 for all three cases, and the graph is omitted.

We generally see large measurement improvement (from 25 to nearly 80 percent), while the increase of σ is very small (from 0.025 to 0). We make the following four observations. First, flow routing achieves the most measurement gain. This is because it utilizes the finest granularity in querying counter

statistics and routing. Second, the measurement improvement for static flowset routing is less than flow routing or dynamic flowset routing. In static flowset routing, some iceberg flows do not belong to the important flowset defined based on μ^* and thus are not routed through the DPI box. Third, due to the tree-like topology, all three cases achieve very close L (average hop count). Finally, the maximum entry number for flow routing is around 700 over time. This number is reduced to around 200 for static flowset routing, but increases to around 300 for dynamic flowset routing. The increased entry number reflects larger average delay since it takes longer to set up entries. In our settings, the average delay is increased by around 0.006 ms/packet for dynamic flowset routing compared to static flowset routing.

Compared to flow routing, the greatest benefit of dynamic flowset routing is the small number of table entries. Compared to static flowset routing, the measurement improvement of dynamic flowset routing is higher. Dynamic flowset routing demonstrates the most common practice of DMR in traffic measurement where no prior iceberg identity of accurate initial measurements of flow sizes are available. It achieves a nice trade-off between measurement improvement and the number of flowset table entries.

Advantage of Rerouting — We next present the advantage of rerouting traffic by the CR/R routing algorithm in Fig. 4. Since L changes little, we omit its graph and present results for κ.

Compared to CR/NR, the measurement improvement is even higher (from 30 to nearly 100 percent), while the increase of σ is still kept at low levels (from 0.03 to 0.005). We make the following three observations. First, CR/R has better measurement improvement than CR/NR. In CR/NR, consistent iceberg flows will not be captured by the DPI box if it initially did not traverse it. Second, similar to Fig. 3, flow routing achieves the largest measurement improvement, while static flowset routing is the worst. Finally, the number of required rule changes, κ, is larger in dynamic flowset routing than in the others. This is because the estimation of iceberg identities is not as accurate as in flow routing. Some non-iceberg flows are also redirected to the DPI box.

Results generally suggest that rerouting traffic is useful to achieve a larger measurement gain, but at the cost of huge churns in terms of number of changes in forwarding rules. Rerouting does not necessarily introduce large σ, since the default 1/R routing can distribute load away from congested links.

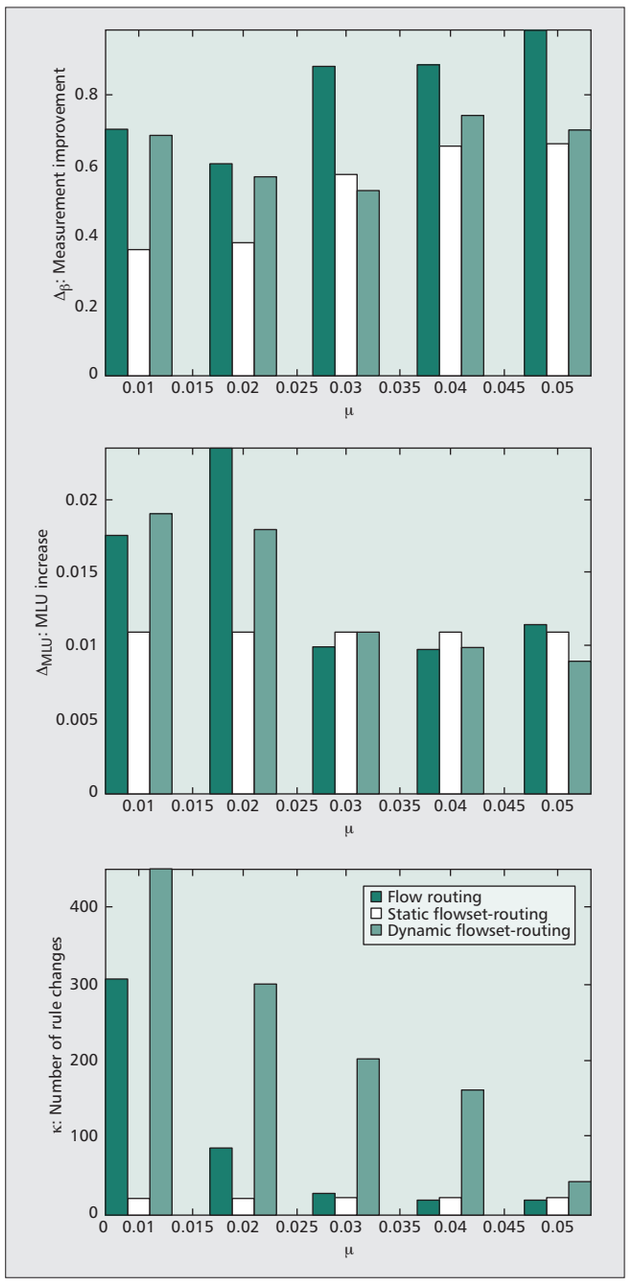

Figure 4. *Advantage of rerouting traffic.*

Concluding Remarks

In this article we propose dynamic measurement-aware routing for network-wide traffic measurement. It overcomes the varying nature of both traffic characteristics and measurement objectives by intelligently routing important traffic sub-populations to some preconfigured monitors. The whole procedure therefore involves initial assessment of flow importance, flowset configuration, and flowset routing with considerations of overall network performances. We discuss the challenges of implementing DMR in practice, build a prototype solution on OpenFlow for global iceberg detection and capture, and evaluate its performance using real traces.

We evaluate three situations: flow routing, static flowset routing, and dynamic flowset routing. They vary from the case of finest granularity of routing control to the case of real practice that can be commonly implemented. We generally see large measurement gain of iceberg detection, while network performance is preserved at decent levels. Our future work includes exploring other measurement applications on OpenFlow and implementing DMR on IP networks.

References

[1] C. Estan *et al.*, "Building a Better NetFlow," *Proc. ACM Sigcomm*, Portland, OR, Sept. 2004.
[2] C. Estan and G. Varghese, "New Directions in Traffic Measurement and Accounting: Focusing on the Elephants, Ignoring the Mice," *ACM Trans. Comp. Sys.*, vol. 21, no. 3, Aug. 2003, pp. 270–313.
[3] J. Mai *et al.*, "Is Sampled Data Sufficient for Anomaly Detection?," *Proc. ACM SIGCOMM IMC*, Rio de Janeiro, Brazil, Oct. 2006.
[4] L. Yuan, C.-N. Chuah, and P. Mohapatra, "ProgME: Towards Programmable Network Measurement," *Proc. ACM SIGCOMM*, Kyoto, Japan, Aug. 2007.
[5] K. Suh *et al.*, "Locating Network Monitors: Complexity, Heuristics and Coverage," *Proc. IEEE INFOCOM*, Miami, FL, Mar. 2005.
[6] S. Raza *et al.*, "Measurouting: A Framework for Routing Assisted Traffic Monitoring," *Proc. IEEE INFOCOM*, San Diego, CA, June 2010.
[7] The Open Flow Switch Consortium, http://www.openflowswitch.org, Aug. 2010.
[8] "Sampling and Filtering Techniques for IP Packet Selection," Internet draft

draft-ietf-psamp-sample-tech-11.txt, work in progress, July 2008.
[9] A. Kumar *et al.*, "Data Streaming Algorithms for Efficient and Accurate Estimation of Flow Size Distribution," *ACM Sigmetrics*, New York, NY, June 2004.
[10] R. N. Mysore *et al.*, "Portland: A Scalable Fault- Tolerant Layer 2 Data Center Network Fabric," *Proc. ACM SIGCOMM*, New York, NY, June 2009.
[11] E.-S. Jung *et al.*, "An Evaluation of In-Advance Bandwidth Scheduling Algorithms for Connection-Oriented Networks," *Proc. Int'l. Symp. Parallel Architectures, Algorithms, and Networks*, 0:133-138, May 2008.
[12] Lbnl enterprise trace repository, http://www.icir.org/enterprise-tracing/, Aug. 2010.

Biographies

GUANYAO HUANG (gyhuang@ucdavis.edu) is a fourth year Ph.D. candidate in the Department of Electrical Computer Engineering, University of California, Davis. His current research focuses on network measurement and anomaly detection. He completed his undergraduate and postgraduate degree from the University of Science and Technology, China.

SAQIB RAZA (sraza@ucdavis.edu) received his B.S. degree in computer science from Lahore University of Management Sciences in 2004, and Ph.D. and M.S. degrees in computer science from the University of California, Davis in 2007 and 2010, respectively. He is a software engineer at the Data Center Switching Technology Group in Cisco Systems, San Jose, California.

SRINI SEETHARAMAN (srini084@gmail.com) is a member of the Clean Slate Laboratory at Stanford University and a senior research scientist with Deutsche Telekom R&D Lab, Los Altos, California. He holds a Ph.D. in computer science from Georgia Institute of Technology and a Master's degree in computer science from The Ohio State University. His research interests include networking architectures and protocols, overlay networks, traffic/network monitoring, and green technologies.

CHEN-NEE CHUAH (chuah@ucdavis.edu) is a professor in electrical and computer engineering at the University of California, Davis. She received her B.S. from Rutgers University, and her M.S. and Ph.D. in electrical engineering and computer Sciences from the University of California, Berkeley. Her research interests lie in the area of communications and computer networks, with emphasis on Internet measurements, network management, cyber-security, online social networks, and vehicular ad hoc networks.