An Approach for Efficient, Accurate, and Timely Estimation of Traffic Matrices

Hongbin Luo, Zhe Chen, Jianbo Cui, and Hongke Zhang
School of Electronic and Information Engineering
Beijing Jiaotong University
Beijing 100044, China
email: {hbluo, 13111036, 12120053, hkzhang}@bjtu.edu.cn

***Abstract*—Network operators desire to obtain accurate evaluations of the traffic matrices of their networks because they are critical inputs to many network functions such as traffic engineering, capacity provisioning and anomaly detection. Under the current Internet architecture, however, it is extremely challenging to precisely measure the traffic between an ingress and egress node pair. In this paper, we argue that a future Internet should make it easy for network operators to be aware of the accurate traffic matrices of their networks in an efficient and timely manner. In particular, we present the requirements for TM estimation and the corresponding implications on the future Internet architecture. Based on these implications, we then present a future Internet architecture that makes it easy to accurately, efficiently, and timely estimate traffic matrices. We also present numerical results to demonstrate the performance of the architecture in estimating traffic matrices.**

Keywords*—future Internet architecture, Traffic matrix estimation, efficiency.

I. Introduction

A traffic matrix (TM) describes the traffic volume (packets/bytes) between an ingress and egress (IE) node pair in a network. Network operators desire to obtain accurate evaluations of the traffic matrices of their networks because they are critical inputs to many aspects of network functions such as traffic engineering, capacity provisioning, and anomaly detection. For example, traffic engineering generally uses the traffic matrix between an IE node pair as input and assigns traffic load among different network paths. Similarly, a sudden increase of traffic volume between an IE node pair may imply a traffic anomaly caused by a denial of service attack.

Under the current Internet architecture, however, it is extremely challenging to accurately, efficiently, and timely measure the TM between IE node pairs. First, the TM in many networks is not directly observable. Instead, it can only be estimated through link load measurements [1]. Second, the collected large amount of per flow/packet information has to be transported to a centralized location for TM estimation [2]. Third, the TM estimation has to be done at very high speed since an Internet service provider (ISP) may forward data packets at a speed of over one Terabits per second (Tbps). While the community has proposed a plethora of methods (e.g., [1-3] and references therein), they *"either do not achieve high level of accuracies, have high operational costs, or are inflexible because their accuracies depend on the unknown traffic volumes"*[2]. In addition, the current Internet faces other serious issues (such as poor security and scalability) that cannot be remedied by incremental changes. Therefore, there are increasing efforts [4-7] in developing "clean-slate" redesigns of the Internet architecture, aiming at rectifying one or more of these problems through non-incremental changes.

In this paper, we argue that a future Internet should make it easy for network operators to be aware of the accurate traffic matrices of their networks in an efficient and timely manner. In particular, we present the requirements for TM estimation and the corresponding implications on the future Internet architecture. Based on these implications, we then present a future Internet architecture that makes it easy to accurately, efficiently, and timely estimate traffic matrices. We also present numerical results to demonstrate the performance of the architecture in estimating traffic matrices.

The rest of the paper is organized as follows. In Section II, we analyze the requirements for TM estimation and their architectural implications. In Section III, we then present a future Internet architecture that makes it easy to accurately, efficiently, and timely estimate traffic matrices. In Section IV, we describe how to estimate traffic matrices of a network under the proposed Internet architecture. In Section V, we present numerical results to demonstrate the performance of the proposed Internet architecture in estimating traffic matrices. Finally, we conclude the paper in Section VI.

II. Requirements and Their Implications

We first present the requirements for TM estimation, which is followed by a description of their implications on the future Internet architecture.

A. Requirements for TM Estimation

We desire that the traffic matrices of a network could be estimated accurately, timely, and efficiently.

978-1-4799-3088-3/14/$31.00 ©2014 IEEE

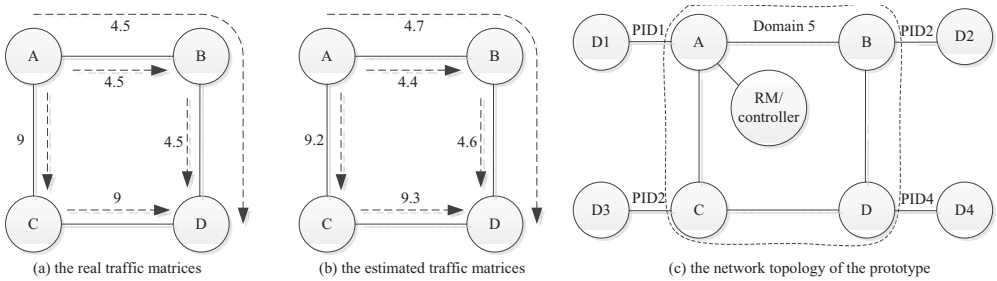

Fig. 1. An example for traffic matrices.

1) Accurately: The emergence of novel technologies such as OpenFlow [8] makes it possible to efficiently use network resources (e.g., bandwidth). This in turn requires that the estimated traffic matrices are accurate since if otherwise, the controller may make wrong decisions when computing paths for flows. Consider the network shown in Fig. 1 as an example, where we assume that each link of the network has a bandwidth of 10 megabytes per second. We also assume that the real traffic matrices are shown in Fig. 1 (a) and the estimated traffic matrices are shown in Fig. 1 (b). In this example, if a flow with node A being the ingress point and node D being the egress point arrives at the network and has a bandwidth requirement of 0.8 megabyte per second, the controller can use either the path A-C-D or the path A-B-D to carry the flow if it knows the real traffic matrices of the network. However, if the controller only knows the estimated but inaccurate traffic matrices, it has to reject the flow since no path has enough bandwidth to carry the flow.

2) Timely: Network managers can make the best use of network resources based on its state-of-the-art network status only when the traffic matrices are estimated in real time. In the above example, if the controller of the network knows the accurate traffic matrices of time t at time $t'(\geq t)$, it cannot compute paths for flows based on the accurate network status during the period $(t,\ t')$. Accordingly, it cannot efficiently use network resources.

3) Efficiently: As stated in Section I, many approaches for estimating traffic matrices relies on a centralized location for correlation. To timely estimate the traffic matrices of a network, however, it requires that the centralized location has very strong computation capability in order to complete the correlation since the average traffic rate of a network may be very high. For example, the average traffic rate of a Tier-1 Internet service provider (ISP) has reached about two Tbps in 2009 [9]. Therefore, it is desirable that the traffic matrices of a network could be efficiently estimated.

B. Architectural Implications

We now analyze the architectural implications of the above requirements for TM estimation. First, the requirement for accurate estimation requires that every byte/packet should be accounted. Nowadays, routers in the network record the number of bytes/packets of flows passing through them and then send the flow information to a centralized location for correlation. However, since some packets may be dropped by some routers along the path from the ingress node to the egress node, the flow information observed at the ingress node and that observed at the egress node may be different. While most approaches use the flow information observed at the egress node for TM estimation, we argue that it is better to use the flow information observed at the ingress node since the role of the network is to forward packets from the ingress node to the egress node with an attempt at dropping as few packets as possible.

Second, the requirements for timely and efficient estimation entail that we should not rely on the correlation of a centralized location because 1) it spends time to send flow information to the centralized location; and 2) correlation is itself time-consuming and inefficient when the traffic rate is high and the network size is large. This implies that the traffic matrices of a network should be estimated in a distributed manner, with each node estimates the traffic matrices from the node to the rest nodes of the network.

These arguments require that a future Internet should make it possible for each ingress node to estimate the traffic matrices from the ingress node to the rest nodes of the network when the ingress node forwards packets to the rest nodes of the network. This implies that *the future Internet architecture should makes it possible for the ingress node of a packet to know the egress node of the packet when the ingress node receives the packet*. By contrast, routers in the current Internet maintain a routing/forwarding table that stores the next hops of IP-prefixes and a router cannot know the egress router of a packet when it receives the packet.

III. A Possible Future Internet Architecture

We will now present a possible future Internet architecture that makes it easy to accurately, efficiently, and timely estimate traffic matrices. The architecture was proposed in [10] and called CoLoR because it couples service location and inter-domain routing while decoupling them from forwarding. Our preliminary results show that CoLoR is promising as it satisfies many requirements of the future Internet including being information-centric, encouraging innovation, and providing efficient support for mobility, multicast, multi-homing,

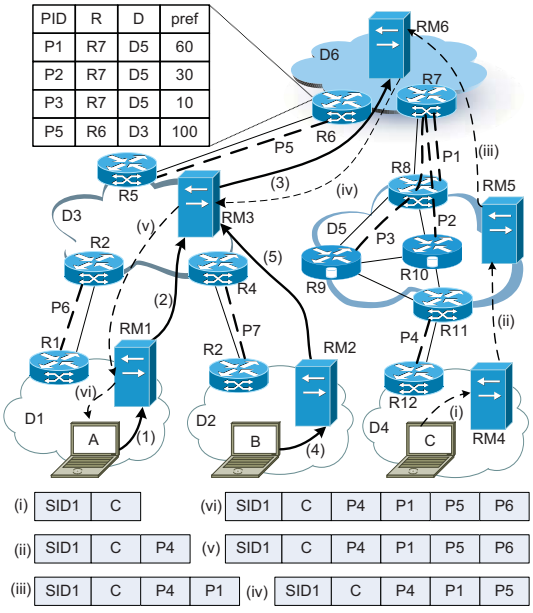

Fig. 2. Illustration for CoLoR.

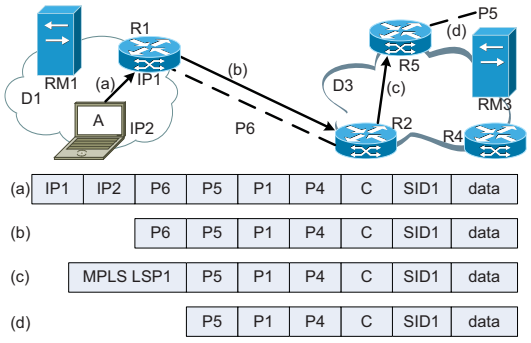

Fig. 3. Illustration for packet forwarding in CoLoR.

and middleboxes [10]. For completeness, we briefly outline CoLoR and refer interested readers to [10] for more details.

As the current Internet, CoLoR assumes that a future Internet is still organized around domains, each of which has a logical resource manager (RM). For brevity, the RM associated with a domain X is denoted by RM_X. Domains have domain-level provider/customer/peer relationships. In addition, RM_X is the provider/customer/peer of RM_Y if domain X is the provider/customer/peer of domain Y in terms of domain-level relationships. CoLoR also assumes that nodes in a domain know how to reach the RM in the same domain.

Domains are free to choose their routing architectures and intra-domain routing mechanisms. For example, domain D1 in Fig. 2 may use IPv4 for intra-domain routing but domain D3 may use IPv6 for intra-domain routing. By contrast, inter-domain routing is based on (virtual) paths, each of which is denoted by a path identifier (PID). In particular, two neighbor domains would negotiate a set of paths that span the two domains. For example, domains D5 and D6 in Fig. 2 negotiate three paths P1, P2, and P3, as indicated by the bold dashed lines in Fig. 2. In addition, the paths negotiated by two domains are not advertised to other domains but are only known by the two domains. Furthermore, for a path that begins at a domain, the RM and routers in the domain maintains the path's end point located at the domain and the domain identifier at which the other end point locates. For example, the inter-domain routing table of R6 is shown at the upper left corner in Fig. 2. Accordingly, the ingress node of a packet can know the egress node of the packet by querying the inter-domain routing table if the packet carries correct PIDs.

For this purpose, CoLoR couples inter-domain routing with service location. In particular, CoLoR names services with persistent service identifiers (SIDs) and stores the reachability information of SIDs at RMs. Specifically, a node holding a copy of a service registers the service's SID to the RM in the same domain, if it is authorized to do so. Depending on the local policy, the RM may also register the service to its parent/peer RMs. Fig. 2 illustrates the register process. When RM3 receives the register message for the SID from RM1, it stores an entry for the SID into its registration table. In addition, it also sends a register message to its parent RM (i.e., RM6). When RM6 receives the register message, it then stores an entry for the SID in its registration table. When RM3 receives a register message for the SID from RM2, it finds that there is an entry for the SID but the register message comes from RM2 instead of RM1. Thus it adds an entry for the SID into its registration table. Note that now RM3 does not send a register message to RM6, since it has registered the SID before.

When a client wants to obtain a service represented by an SID, it sends out a GET message to its local RM. The GET message should contain the SID and the client's node identifier (NID). If the RM cannot find an entry for the SID, it firstly chooses a parent RM based on its local policy and then chooses a path destined to the corresponding parent domain. The RM then appends the PID of the chosen path onto the GET message and sends it to the chosen parent RM. If the RM finds at least one entry for the SID, it chooses an entry based on its local policy (e.g., traffic engineering). If the node contained in the entry is in the same domain with the RM, the RM directly forwards the GET message to the node by using the routing mechanism in that domain. If the node is in another domain, the RM chooses a path towards that domain, based on its local policy. Afterwards, the RM appends the PID of the chosen path onto the GET message and forwards the GET message to the node. This way, the GET message will be forwarded to the node holding the desired service. The steps (i) - (iv) in Fig. 2 illustrate how a GET message is forwarded to a node hosting a desired service.

When the node holding the desired service receives the GET message, it encapsulates the inter-domain paths into

headers of data packets for the service. Packets for the service will be forwarded domain-by-domain, until they arrive at the client. Within every domain, packets are forwarded by using the local approach chosen by the domain. On the other hand, when passing through two domains, they are forwarded by using PIDs carried in the packet header. Fig. 3 illustrates how data packets are forwarded in CoLoR, assuming that domains D1 and D3 in Fig. 2 use IP and MPLS for local routing, respectively. When node A receives the GET message, it firstly encapsulates the data with a header that contains SID1, the client's NID (i.e., C), and the PIDs. Since domain D1 uses IP for local routing, node A then encapsulates the packet with an outer IP header whose source and destination addresses are IP2 and IP1, respectively, as illustrated by (a) in Fig. 3. The packet will be forwarded to node R1 by IP routing. R1 then strips out the outer IP header and sends the packet to path P6, as illustrated by (b) in Fig. 3. When node R2 receives the packet, it stripes out the PID P6 and finds that it should forward the packet to path P5. As a result, it queries its inter-domain routing table and knows that the egress node of the packet is node R2. After that, it encapsulates the packet with an outer MPLS header that contains the LSP between node R2 and node R5 (i.e., LSP1), since domain D3 uses MPLS for local routing, as illustrated by (c) in Fig. 3. When R5 receives the packet, it strips out the MPLS header and sends the packet to path P5, as illustrated by (d) in Fig. 3. This way, the packet will ultimately be sent to the domain that the destination locates (i.e., domain D4 in Fig. 2). Domain D4 then uses the node identifier carried in the packet header to forward the packet to node C.

IV. Traffic Matrix Estimation in CoLoR

We now show how CoLoR makes it easy to accurately, efficiently, and timely estimate traffic matrices. Recall that in CoLoR, each node in a domain maintains a routing table that records for every PID in the domain, the neighboring domain that the domain connects through the PID, and the border router that the PID originates at the domain. Thus when a router in a domain receives a packet, it knows either a PID if the packet is destined to another domain, or an NID if the destination node is in the same domain. In the former case, the router knows the egress border router of the packet by querying its routing table during the packet forwarding process. In the latter case, the router knows the NID of the destination. Accordingly, to obtain the traffic matrix from the router to any other node in the network, the router only needs to count these packets (or the packet sizes) that contain the PID (or NID).

In the example network shown in Fig. 2, when the border router R5 in domain D3 receives packets destined to domain D1, it can find the PID P6 in these packets. To estimate the number of packets from R5 to R2 during a period, R5 simply sums the number of packets that contain the PID P6 during that period when it forwards these packets. Similarly, to estimate the number of bytes from R5 to R2, R5 only sums the number of bytes of packets containing the PID P6 when it forwards these packets. This way, R5 accurately estimates the traffic matrix from R5 to R2, without any prior knowledge or assumption. Similarly, R5 is able to accurately estimate the traffic matrix from R5 to R4.

To accurately estimate the traffic matrices in a network, the RM in the network simply informs every router in the network to estimate the traffic matrices from the router to the rest routers during a given period. When the routers complete the estimation, they report the estimated traffic matrices to the RM. Accordingly, the RM knows the traffic matrices in the whole network.

The TM estimation in CoLoR has at least three benefits over that in the current Internet. First, the traffic matrices in a network are efficiently estimated by the routers in parallel. This is in sharp contrast to the TM estimation in the current Internet, where the collected large amount of per flow/packet information has to be shipped to a centralized location for correlation [2].

Second, the traffic matrices can be timely estimated in CoLoR, since a router is able to estimate the traffic matrix at line speed when it forwards data packets. By contrast, it is difficult to timely estimate traffic matrices at very high speed in the current Internet [1].

Third, CoLoR makes it easy to accurately estimate traffic matrices without making any assumption, because routers estimate traffic matrices when they forward data packets and every byte/packet is counted. By contrast, many approaches proposed for estimation traffic matrices in the current Internet are inflexible because their accuracies depend on the unknown traffic volumes [2].

We note that, for a given router, the estimation of traffic matrices from the router to other routers only slightly increase its workload, since routers in the current Internet also count the number of packets/bytes they forwarded. While a router needs to maintain an entry for every PID, the number of PIDs in a network is very limited. For example, as of October 14, 2013, the largest autonomous system (AS) in terms of AS degrees has a degree of 4,131 [11]. If an AS has two PIDs with each of its neighbor ASes, the maximal number of PIDs that an AS has is less than 10,000. To estimate the traffic matrix from an ingress router to an egress router (corresponding to a PID), an ingress router maintains an entry that contains three fields: the PID, the number of packets and the number of bytes to the PID. Assuming that each of the three fields has a length of 32 bits, an entry requires 96 bits memory space. Thus an ingress router needs at most 120,000 (= 10,000*96/8) bytes memory space, which is fairly small when compared with the memory size of commodity routers.

V. Performance Evaluation

We will now present numerical results to show the performance of CoLoR in estimating traffic matrices. In particular, we compare the traffic matrix estimation in CoLoR with that in OpenFlow because, when compared with the current

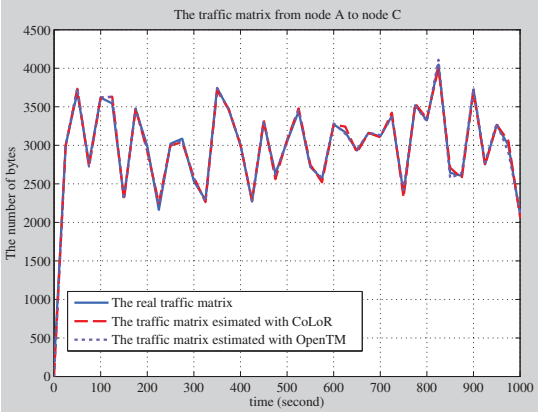

Fig. 4. The real and estimated traffic matrix in the first scenario.

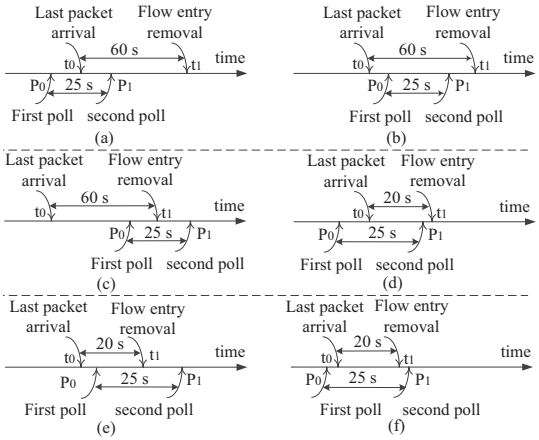

Fig. 5. Illustration for the effectiveness of OpenTM.

Internet, OpenFlow makes it easier to accurately estimate the traffic matrices of a network [8]. Specifically, we use the OpenTM approach proposed in [12] to estimate the traffic matrices in OpenFlow.

In OpenTM, the controller of a network records all active flows in the network. For each flow, the controller computes the flow path based on the routing information, and periodically polls flow byte and packet-count counters from switches along the flow path. By subtracting the flow byte (or packets) of two subsequent polls, the controller then knows the number of bytes/packets of the flow during the past period (e.g., five seconds). To estimate the traffic matrix from an ingress switch to an egress switch, the controller simply sums up all flows originated from the same ingress switch to the same egress switch. By computing the traffic matrix of every ingress and egress switch pair, the controller is able to know the traffic matrices of the network. We built a prototype that comprises four routers/switches, four end hosts (i.e., D1, D2, D3, D4), a controller (or an RM), as illustrated by Fig. 1 (c). For CoLoR, we assume that each end host behaves as a domain, and the four routers comprise a domain (i.e., Domain 5), as illustrated by the dotted circle in Fig. 1 (c). The PIDs between Domain 5 and the other four domains are shown in Fig. 1 (c).

When running the prototype, we let host D1 send data packets belonging to different flows to host D3. We generate two scenarios of flows. In the first scenario, we randomly generate nine flows whose durations follow the uniform distribution between 30 to 50 seconds. When a flow terminates, we generate another flow after a period whose duration also follows the uniform distribution between 20 to 40 seconds. Each flow sends out ten data packets per second. The size of a data packet is also fixed to 60 bytes. In this scenario, an OpenFlow controller polls the switches once every 25 seconds but the default value for a switch to remove a flow entry from its flow table is set to be 60 seconds. Similarly, a router in CoLoR estimates the traffic matrix from node A and node C, and reports the estimated traffic matrix to the RM once every 25 seconds.

Fig. 4 shows the real traffic matrix and the estimated traffic matrices by using the OpenTM approach and the CoLoR approach in the first scenario. From this figure, we can observe that the estimated traffic matrices match the real traffic matrix very well. Note that, in this scenario, the controller in OpenTM polls the switches once every 25 second but the default value for a switch to remove a flow entry from its flow table is 60 seconds. Accordingly, OpenTM is able to accurately estimate the traffic matrix. We explain this by using the example shown in Fig. 5. In particular, we consider the case that the last packet of a flow arrives at a switch but the flow entry is not removed from the flow table of the switch. For ease of presentation, we denote the time that the last packet arrives at the switch by t_0, and the time that the corresponding flow entry is removed from the flow table of the switch by t_1. In addition, we denote the time that two consecutive polls arrive at the switch by P_0 and P_1. Since the interval between the two consecutive polls is less than the default value that a flow entry is removed from the flow table, there are three possible cases that may occur. In the first one, the first poll arrives before the last packet arrives at the switch, as illustrated by Fig. 5 (a). In this case, OpenTM is able to count the number of bytes forwarded from time P_0 to t_0. In the second case, the first poll arrives after the last packet arrives at the switch, and the second poll arrives before the flow entry is removed from the flow table, as illustrated by Fig. 5 (b). In this case, no packet is forwarded during the two consecutive polls. In the third case, the first poll and the second poll arrive at the switch before and after the flow entry is removed, respectively, as illustrated by Fig. 5 (c). Now, the controller finds that the flow entry is removed and no packets/bytes are counted. In all the three cases, OpenTM is able to accurately estimate the traffic matric if the duration between two consecutive polls is less than the default value for a switch to remove a flow entry from its flow table.

However, if the duration between two consecutive polls is larger than the default value for a switch to remove a flow entry, OpenTM may not be able to accurately estimate the traffic matrices. To evaluate the performance of OpenTM in

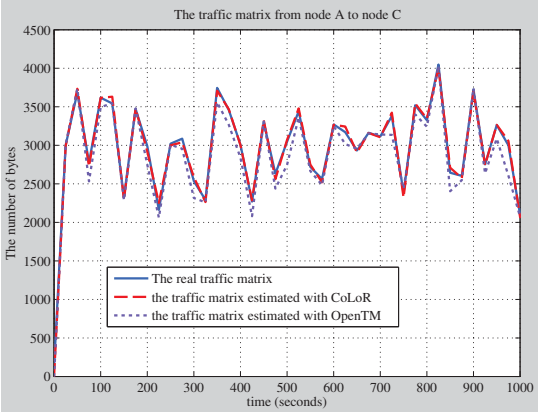

Fig. 6. The real and estimated traffic matrix in the second scenario.

estimating traffic matrices in this case, we consider the second scenario where the default value for a switch to remove a flow entry from its flow table is changed from 60 seconds to be 20 seconds but the other parameters keep unchanged.

Fig. 6 shows the real traffic matrix and the estimated traffic matrices by using CoLoR and OpenTM in the second scenario. We can observe from Fig. 6 that the estimated traffic matrix with CoLoR still matches very well with the real traffic matrix. However, the traffic matrix estimated with OpenTM does not match with the real traffic matrix. We explain this through the examples shown in Fig. 5 (d) - (f). In Fig. 5 (d), the first poll arrives at the switch before the last packet arrives; on the other hand, the second poll arrives at the switch after the last packet arrives but before the flow entry is removed from the flow table. In this case, OpenTM can count the number of packets/bytes forwarded during P_0 and t_0. In Fig. 5 (e), the first poll arrives at the switch after the last packet arrives at the switch but before the flow entry is removed from the flow table. On the other hand, the second poll arrives at the switch after the flow entry is removed from the flow table. Now, no packets are forwarded during the two consecutive polls. In Fig. 5 (f), the first poll arrives at the switch before the last packet arrives, but the second poll arrives at the switch after the flow entry is removed. In this case, the controller cannot count the number of packets/ bytes forwarded during P_0 and t_0. Thus, OpenTM cannot accurately estimate the traffic matrix if the duration between two consecutive polls is larger than the default value for a switch to remove a flow entry.

It is also worthy of noting that, as stated in [12], OpenTM cannot be used to large networks where the number of flows is huge. In addition, to estimate the traffic matrix between an IE pair in OpenTM, the controller needs to compute the ingress and egress points of a flow, and to sum up all the flows passing through the IE pair. By contrast, the traffic matrices in a large network are still easy to estimate in CoLoR since they are estimated in parallel by routers in the network.

VI. Conclusion

We have analyzed the requirements for traffic matrix estimation and their architectural implications on a future Internet. We have also presented a possible approach called CoLoR and described how CoLoR makes it easy to efficiently, accurately, and timely estimate traffic matrices. The results show that CoLoR outperforms OpenTM in estimating traffic matrices.

Acknowledgment

This work was supported in part by the 973 Program under Grant No. 2013CB329100, in part by NSFC under Grant No. 62171200 and 62132017, in part by the Ph.D. Programs Foundation of the Ministry of Education of China under Grant No. 20130009110014, in part by the Program for New Century Excellent Talents in University ("NCET") under Grant No. NCET-12-0767, and in part by the Fundamental Research Funds for the Central Universities under Grant No. 2014JBM011.

References

[1] Y. Zhang, M. Roughan, W. Willinger, L. Qiu,"Spatio-temporal compressive sensing and internet traffic matrices," in *Proc. ACM SIGCOMM'09*, August 2009, Barcelona, Spain.

[2] J. Cao, A. Chen, and T. Bu, "A quasi-likelihood approach for accurate traffic matrix estimation in a high speed network," in *Proc. IEEE INFOCOM' 08*, April 2008, Phoenix, AZ, USA.

[3] A. Callado, C. Kamienski, G. Szabo, B. P. Gero, J. Kelner, S. Fernandes, and D. Sadok, "A survey on Internet traffic identification," *IEEE Communications Surveys and Tutorials*, vol. 11, no. 3, Third Quarter 2009, pp. 37 - 52.

[4] T. Koponen, M. Chawla, B. - G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, I. Stoica, "A data-oriented (and beyond) network architecture," In *Proc. ACM SIGCOMM'07*, Aug. 2007, Kyoto, Japan.

[5] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, R. L. Braynard, "Networking named content," In *Proc. ACM CoNEXT'09*, Dec. 2009, Rome, Italy.

[6] D. Trossen, M. Sarela, K. Sollins, "Arguments for an information-centric internetworking architecture," *ACM SIGCOMM CCR*, vol. 40, no. 2, April 2010, pp. 27 - 33.

[7] MobilityFirst Future Internet Architecture Project Overview. http://mobilityfirst.winlab.rutgers.edu/.

[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM CCR*, vol. 38, no. 2, April 2008, pp. 69 - 74.

[9] C. Labovitz, S. lekel-Johnson, D. McPherson, J. Oberheide, F. Jahanian, "Internet inter-domain traffic," in *Proc. SIGCOMM' 10*, Aug. 2010, New Delhi, India.

[10] H. Luo, Z. Chen, J. Cui, H. Zhang, M. Zukerman, and C. Qiao, "CoLoR: An information-centric Internet architecture for innovation," *IEEE Network Magazine*, to appear.

[11] BGP Peer Report. http://bgp.he.net/report/peers.

[12] A. Tootoonchian, M. Ghobadi, Y. Ganjali, "OpenTM: traffic matrix estimator for OpenFlow networks," In *Proc. 11th International Conference on Passive and Active Measurement (PAM'10)*, April 2010, Zurich, Switzerland, pp. 201 - 210.