Reliable Segment Routing

(Invited Paper)

Alessio Giorgetti, Andrea Sgambelluri, Francesco Paolucci, and Piero Castoldi
Scuola Superiore Sant'Anna
Pisa, Italy
Email: a.giorgetti@sssup.it

***Abstract*—Segment Routing (SR) is a novel traffic engineering technique compatible with traditional MPLS data plane. SR relies on label stacking to steer traffic flows throughout the network. Signaling protocol is not required, thus control plane operation is greatly simplified. SR can also be exploited upon network failures to promptly perform traffic recovery and, subsequently, to optimize the recovered traffic for avoiding network congestion.**

This study proposes a procedure to dynamically recover the traffic flows disrupted by a single failure in segment routing networks. The proposed procedure is evaluated in several network topologies to estimate the complexity of the required operations.

I. Introduction

Segment Routing (SR) has been recently proposed as an innovative technique to provide traffic engineering (TE) by simplifying control plane operation [1], [2]. SR relies on source routing and can be easily deployed in Multiprotocol Label Switching (MPLS) networks. Indeed, according to SR, packet flows are enforced through a given path by applying, at the ingress node, a specifically designed stack of segment identifiers (i.e., labels) fully compatible with the MPLS data plane.

The stack of labels is named *segment list*. Only the top label in the list is processed during packet forwarding. In particular, each packet is forwarded along the shortest path toward the network element represented by the top label. For instance, a label can represent an Interior Gateway Protocol prefix identifying a specific router (i.e., IGP-Node Segment [1]). Unlike traditional MPLS networks, SR maintains per-flow state only at the ingress node, where the segment list is applied. Therefore, no signaling protocol (e.g., Reservation Protocol with traffic engineering extensions - RSVP-TE) is required to populate the forwarding table of transit nodes. In this way, a simplified control plane is employed, just relying on an IGP properly extended to advertise ythe segment identifiers [3]. Thus, scalability of transit nodes is greatly improved, since MPLS Label Switch Paths (LSPs) state information is not required.

Several interesting SR use cases have been recently proposed. As an example, SR can be effectively used to steer traffic flows on paths characterized by low latency values and to avoid link congestion using a limited number of tunnels [4]. Moreover, segment routing can be effectively used upon network failures to promptly perform traffic recovery and, subsequently, to optimize the recovered traffic for avoiding network congestion that may occur on the faulted network topology.

However, SR may suffer from other potential issues [5]. Indeed, deployed MPLS equipments typically support a limited number of stacked labels. Therefore, the segment list depth may be constrained to a limited value to guarantee backward compatibility. Moreover, adding a segment list to each packet, introduces additional packet overhead.

Even though standardization is rapidly evolving and the main network equipment vendors are currently involved in the implementation phase [6], there has been a limited research work on SR within the academic community; especially, the possible SR application during recovery phase has not yet been investigated.

Authors of [7] propose to combine the benefits of SR with those of a software defined networking (SDN) architecture [4]. The works in [5], [8] propose a SR implementation for Carrier Ethernet networks including both a testbed evaluation and a simulation study. The proposed solution is able to reduce the required segment list depth but implies integrations of the segment list at some intermediate nodes (i.e., the swap nodes). Our previous work in [9] proposes an algorithm to compute the segment list using an auxiliary network graph. Finally, the works in [10]–[12] propose two experimental implementations of SR respectively based on an OpenFlow-based controller and a Path Computation Element (PCE) based controller.

This study proposes a segment routing procedure to dynamically recover traffic flows disrupted by link or node failures. Within the proposed procedure the central controller is not involved upon failure occurrence so that the traffic is locally recovered at the node detecting the failure guaranteeing short recovery time. In a second phase the controller may be used to optimize the traffic flows on the faulted topology to avoid link congestion. Several network topologies have been considered to evaluate the proposed procedure and the obtained results demonstrated that, in most of the cases, traffic recovery is performed inserting no more than two new labels in the segment list.

II. Segment Routing operation

Segment Routing typically requires the utilization of a centralized controller, such as a PCE [10], [13] or an SDN controller [4], [7], [14], [15]. When a new traffic flow has to be established, a request is issued to the controller that computes the path and encodes the computed path using a segment list.

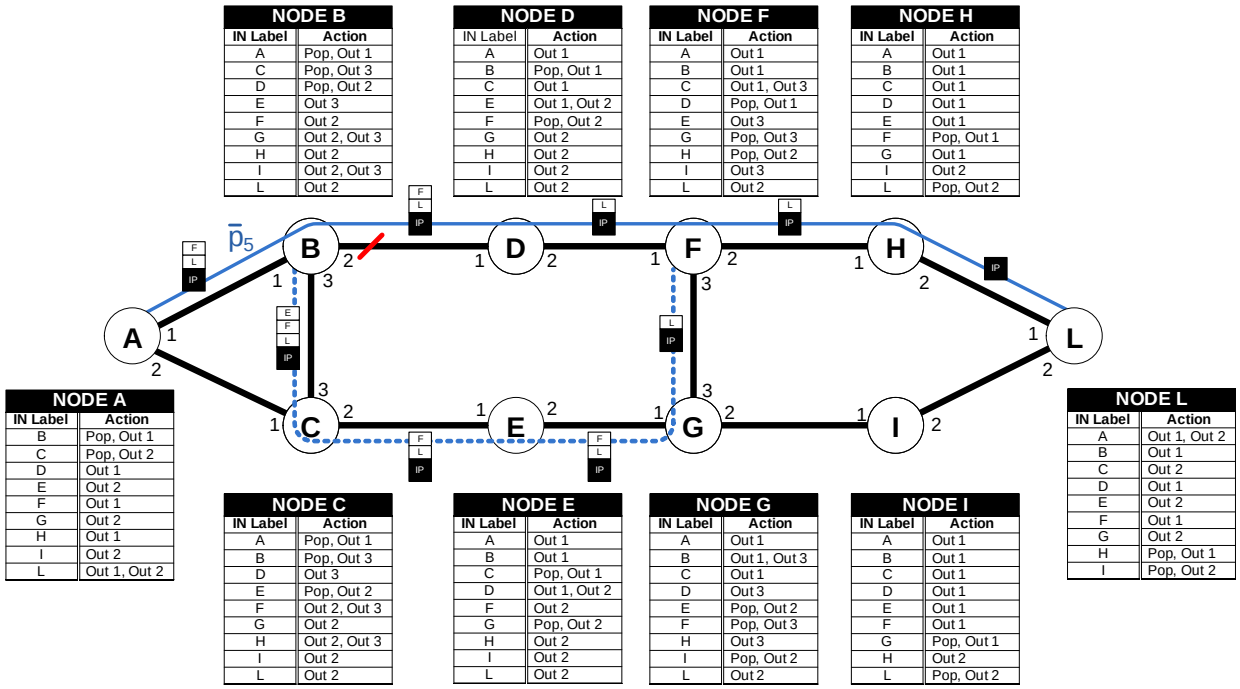

Fig. 1. Test network topology detailing Segment Routing examples. The forwarding table is reported for each node. The path $\bar{p}_5$ is illustrated with solid line, the backup path for $\bar{p}_5$ is illustrated with dashed line in case of failure of link $B - D$. Also the segment list of packets traversing $\bar{p}_5$ and its backup path is illustrated hop by hop, where node B pushes the label E along the backup path.

Fig. 1 explains the basic segment routing operations. Specifically, the figure illustrates a network where all the forwarding tables are detailed assuming that an IGP is used to advertise the node identifiers, as proposed in [3].

If a new traffic request arrives from node A to node I, the controller computes the path $\bar{p}_1 = \{A, C, E, G, I\}$. Since $\bar{p}_1$ is the *unique shortest path* from A to I, the segment list $\overline{SL}^{\bar{p}_1}$ encoding $\bar{p}_1$ only includes one label (i.e., $\overline{SL}^{\bar{p}_1} = \{I\}$). The packets are then forwarded along $\bar{p}_1$ without modifying the segment list up to node G where the label I is popped (i.e., penultimate hop popping) and the packet is forwarded to node I.

Alternatively, if the controller computes the path $\bar{p}_2 = \{A, B, D, F, G, I\}$ a more complex segment list is required because $\bar{p}_2$ is not the unique shortest path to node I. A possible segment list is $\overline{SL}^{\bar{p}_2} = \{F, I\}$. Therefore, packets are forwarded up to node D without modification to the segment list. At node D the label F is popped, then node F receives the packets with the single label I and forwards them through the unique shortest path toward node I. At node G, the label I is popped and the packet is forwarded to node I.

A. Equal Cost Multi Path aware routing

SR natively implements Equal Cost Multi Path (ECMP) aware routing, i.e., in case of multiple shortest paths toward the destination the traffic is load balanced on a per-flow basis. For instance, in Fig.1 traffic from C to F and labeled with the single label F will be load balanced on the two paths $\bar{p}_3 = \{C, E, G, F\}$ and $\bar{p}_4 = \{C, B, D, F\}$.

NODE D

IN Label	Action	Backup Action
A	Out 1	Push G, Out 2
B	Pop, Out 1	Push E, Out 2
C	Out 1	Push G, Out 2
E	Out 1, Out 2	Out 2, Out 1
F	Pop, Out 2	Push E, Out 1
G	Out 2	Push C, Out 1
H	Out 2	Push E, Out 1
I	Out 2	Push C, Out 1
L	Out 2	Push C, Out 1

(a)

NODE B

IN Label	Action	Backup Action
A	Pop, Out 1	Out 3
C	Pop, Out 3	Out 1
D	Pop, Out 2	Push G, Out 3
E	Out 3	Push F, Out 2
F	Out 2	Push E, Out 3
G	Out 2, Out 3	Out 3, Out 2
H	Out 2	Out 3
I	Out 2, Out 3	Out 3, Out 2
L	Out 2	Out 3

(b)

Fig. 2. Link failure recovery - forwarding tables including backup actions: (a) node D, (b) node B.

NODE D

IN Label	Action	Backup Action
A	Out 1	Push G, Out 2
B	Pop, Out 1	Pop
C	Out 1	Push G, Out 2
E	Out 1, Out 2	Out 2, Out 1
F	Pop, Out 2	Pop
G	Out 2	Push C, Out 1
H	Out 2	Push E, Out 1
I	Out 2	Push C, Out 1
L	Out 2	Push C, Out 1

(a)

NODE B

IN Label	Action	Backup Action
A	Pop, Out 1	Pop
C	Pop, Out 3	Pop
D	Pop, Out 2	Pop
E	Out 3	Push F, Out 2
F	Out 2	Push E, Out 3
G	Out 2, Out 3	Out 3, Out 2
H	Out 2	Out 3
I	Out 2, Out 3	Out 3, Out 2
L	Out 2	Out 3

(b)

Fig. 3. Node failure recovery - forwarding tables including backup actions: (a) node D, (b) node B.

To avoid ECMP load balancing, i.e., a strict path is desired, additional labels are required in the segment list. For instance, if the target path is $\bar{p}_3$ two labels are required in the segment list to discriminate between the two shortest paths (e.g., $\overline{SL}^{\bar{p}_3} = \{G, F\}$).

TABLE I
SEGMENT ROUTING RECOVERY RESULTS SUMMARY

	N	L	Label to push link ECMP		Label to push link STRICT		Label to push node ECMP		Label to push node STRICT	
			Avg.	Max	Avg.	Max	Avg.	Max	Avg.	Max
PanEU	27	55	0.01	1	0.54	2	0.19	2	0.78	3
NSF	14	21	0.28	1	0.48	2	0.2	1	0.42	1
MultiD	75	145	0.07	1	0.79	4	0.27	2	1.05	4
BRITE	100	200	0.05	1	0.63	4	0.09	2	0.64	4
BRITE	150	300	0.07	1	0.6	4	0.12	2	0.62	4
Ring	27	27	0.92	1	0.92	1	0.92	1	0.92	1
Grid	25	40	0.11	1	0.80	2	0.08	1	0.73	2

B. Label stack computation

The segment list used for encoding a pre-determined path (i.e., the *target path*) is computed at the controller after the computation of the path (i.e., the target path) for each new traffic flow. In this paper the Segment Routing Reverse (*SR-R*) algorithm proposed in [16] is used.

First, the SR-R algorithm inserts the destination node in the segment list. Then the algorithm generates a number of target segments by navigating the computed target path in the backward direction from the destination node up to the source node. This way, the first evaluated target segment includes the last two nodes of the target path, the second one includes the last three nodes and so forth. If the generated target segment is a unique shortest path, an additional node is included from the target path. Otherwise, the source node of the previous target segment is included in the segment list, and the algorithm iterates starting from the node inserted in the segment list.

III. SEGMENT ROUTING RECOVERY PROCEDURES

This paper proposes a procedure to implement fast recovery in Segment Routing networks. The proposed procedure does not involve the controller in the traffic recovery upon failure occurrence. Specifically, the forwarding table of each network node is properly configured during the network initialization phase so that when a node physically detects a failure of a connected interface it is able to deviate the traffic on a backup path. This way, not only the controller is not involved upon failure occurrence, but also the node detecting the failure does not require to perform additional path computations to redirect the traffic on the backup path.

The proposed procedure can be implemented using an SDN controller and OpenFlow-based switches where the forwarding table can include actions encoded in the form of groups that natively supports the monitoring of the interface state and a backup list of actions that is applied if primary forwarding interface is down [14], [17]. Alternatively, it can be implemented using a PCE acting as centralized controller of an IP/MPLS network [12].

The next two subsections specify the proposed procedure in two different configurations, respectively designed to consider link failures and node failures.

A. Link failure

Traffic flows are re-routed from the node detecting the failure up to the node indicated in the next label of the segment list. Depending on the traffic flows the next label can indicate the next hop in the network topology, a farther node or even the destination node of the traffic flow. Therefore, the proposed approach is an intermediate solution between link protection (e.g., standard MPLS Fast Reroute, FRR) and path protection.

As an example, consider the network in Fig. 1 and a traffic flow traversing the path $\bar{p}_5 = \{A, B, D, F, H, L\}$ (solid line in Fig. 1) encoded with the segment list $\overline{SL}^{\bar{p}_5} = \{F, L\}$. In case of failure of link $D-F$ node D detects the failure and the next label in the segment list is F that is a proper next hop in the network topology. In this case the backup path is computed from D to F excluding the disrupted link from the network topology, i.e., $D - B - C - E - G - F$ and the proposed method acts as a traditional link protection strategy. In case of failure of link $B - D$ node B detects the failure and the next label in the segment list is still F. In this case F is not a next hop in the network topology. The backup path is computed from B to F, i.e., $B-C-E-G-F$ (dashed line in Fig. 1). Finally, considering path $\bar{p}_1 = \{A, C, E, G, I\}$ encoded with the segment list $\overline{SL}^{\bar{p}_1} = \{I\}$; in case of failure of link $A-C$ the next label is I and the backup path is computed as in path protection from A to the destination node I.

Specifically, during the network initialization phase the controller considers for each node the possible detection of a failure on each local interface. Thus, for each entry in the forwarding table a list of backup actions are programmed. Considering path $\bar{p}_2 = \{A, B, D, F, G, I\}$, in case of failure of link $D - F$ the backup path computed at the controller is $D-B-C-E-G-F$ thus the backup interface to be used at node D is port 1. However, if packets are forwarded on port 1 without adding labels in the segment list they enter node B with label F, and may generate packet looping. Indeed, port 2 is the port indicated as output port in the forwarding table for packets labeled with F. Therefore, before forwarding packets on port 1, node D is required to perform a push operation adding a label to steer the packets along the computed backup path. In this example, it is sufficient that the single label E is pushed, so that node B will forward the packets using port 3 toward node C, see Fig. 2(b).

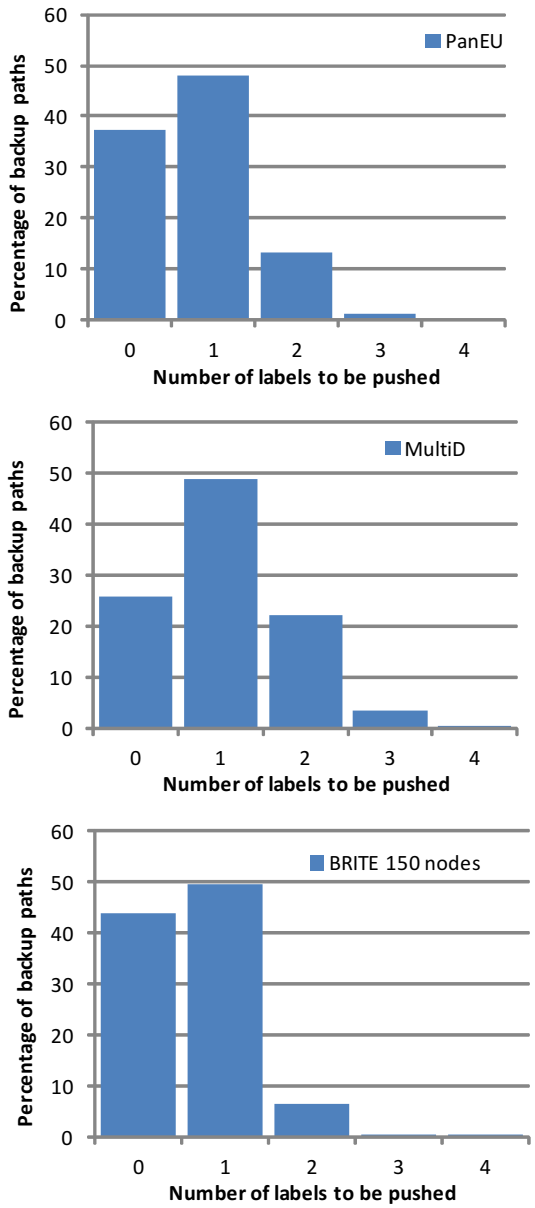

Fig. 4. Percentage of backup paths requiring a specific number of pushed labels (a) PanEU; (b) MultiD; (c) BRITE topologies with 150 nodes.

Following the aforementioned procedure Fig. 2 illustrates the forwarding tables of nodes D and B, where the backup actions include for each entry a list of labels to be pushed and the backup interface to forward the re-routed packets. Specifically, the figure illustrates that at node D almost all the backup entries require a push operation, whereas at node B only 3 entries out of 9 require a push operation. As an example, at node B, if interface 2 is down the traffic directed toward node L can be simply redirected to node C without requiring push operations. Indeed, the default action at node C for the traffic labeled with L will forward it on the desired backup path, i.e., $B - C - E - G - F$.

B. Node failure

A modification to the scheme described in the previous section is required to recover from node failures. First, the backup paths are computed at the controller by removing from the topology the faulted node and all the locally connected links. Then the segment list corresponding to the computed backup path is computed with the algorithm described in Sec. II-B assuming the network without failures. Finally, the backup actions in the forwarding table of each network node are initialized such that if the next label indicates an adjacent node (i.e., the node affected by the failure) a backup path directed to the next next label is enforced, thus a pop operation is applied and the packet is processed depending on the value of the next next label.

Applying this procedure Fig. 3, illustrates the forwarding tables of nodes D and B, where the backup actions can also include pop operations.

IV. Numerical Results

This section evaluates the two proposed recovery procedures considering link and node failures, on several network topologies in terms of labels to be pushed in the segment list at the node detecting the failure. For each procedure two versions are considered: in the first version (i.e., ECMP) the deviated traffic is allowed to reach the destination exploiting load balancing along all the available shortest paths, in the second version (i.e., STRICT) the deviated traffic is enforced along a single backup path computed at the controller during the network initialization phase.

The considered topologies can be divided in three groups: (i) real topologies; (ii) topologies generated with BRITE [18]; (iii) regular topologies (i.e., grid and ring). Tab. I summarizes the network topology details and the obtained results. Specifically, for each topology, Tab. I reports: the number of nodes N; the number of bidirectional links L; the average and the maximum number of labels to be pushed in the stack for each considered scheme. The detailed representation of PanEU, MultiD and NSF networks can be found in [13] and [19].

Tab. I shows that the average number of labels to be pushed is less than 1 in all the considered cases, except for node failure STRICT scheme in the MultiD topology. Therefore, in most of the case no additional labels are required to perform the traffic recovery. A significant exception is the ring topology where in almost all the backup paths one label is sufficient to allow the traffic recovery. Moreover, it is worth to note that the schemes considering node failures require a slight higher number of labels with respect to the scheme considering only link failures; if a strict backup path is required the number of required labels increases and in such cases up to 4 labels may be required.

Finally, Fig. 4 illustrates the percentage of backup paths requiring the pushing of a specific number of labels considering PanEU, MultiD, and 150 nodes BRITE topologies and the STRICT scheme supporting node failures. For all the distributions we can conclude that the maximum value is assumed only for a very few number of backup paths. As an

example, for the three considered topologies, more than 90% of the backup paths requires the pushing of two or less labels.

V. Conclusions

This paper introduced two procedures for effective traffic recovery in segment routing networks considering link and node failures, respectively. The schemes were applied on a number of network topologies to evaluate the number of new labels required at the node detecting the failure to steer the trafffic on the backup path. Results demonstrated that in most of the cases no more than two labels are required.

Acknowledgment

This work was partially funded by the European Community's through the PACE project.

References

[1] C. Filsfils *et al.*, "Segment routing architecture," *draft-filsfils-spring-segment-routing*, 2014.

[2] ——, "Segment routing with MPLS data plane," *draft-filsfils-spring-segment-routing-mpls*, 2014.

[3] P. Psenak *et al.*, "OSPF extensions for segment routing," *draft-psenak-ospf-segment-routing-extensions-05*, 2014.

[4] G. Swallow, "From tag switching to SDN and segment routing MPLS: an enduring architecture," in *Proc. MPLS SDN World Congress*, Mar. 2014.

[5] S. Bidkar, A. Gumaste, and A. Somani, "A scalable framework for segment routing in service provider networks: The omnipresent Ethernet approach," in *Proc. HPSR*, Jul. 2014.

[6] C. Filsfils, "Segment routing: Update and future evolution," in *Proc. MPLS SDN World Congress*, Mar. 2014.

[7] D. Cai, A. Wielosz, and S. Wei, "Evolve carrier Ethernet architecture with SDN and segment routing," in *Proc. WoWMoM*, Jun. 2014.

[8] S. Bidkar, A. Gumaste, P. Ghodasara, S. Hote, A. Kushwaha, G. Patil, S. Sonnis, R. Ambasta, B. Nayak, and P. Agrawal, "Field trial of a software defined network (SDN) using carrier ethernet and segment routing in a tier-1 provider," in *Globecom Conf.*, Dec. 2014.

[9] F. Lazzeri, G. Bruno, J. Nijhof, A. Giorgetti, and P. Castoldi, "Efficient label encoding in segment-routing enabled optical networks," in *Proc. ONDM*, May 2015.

[10] A. Sgambelluri, F. Paolucci, A. Giorgetti, F. Cugini, and P. Castoldi, "SDN and PCE implementations for segment routing," in *Proc. NOC*, Jul. 2015.

[11] A. Sgambelluri, A. Giorgetti, F. Cugini, G. Bruno, F. Lazzeri, and P. Castoldi, "First demonstration of SDN-based segment routing in multi-layer networks," in *Optical Fiber Communication (OFC) Conference*, March 2015, pp. Th1A–5.

[12] A. Sgambelluri, F. Paolucci, A. Giorgetti, F. Cugini, and P. Castoldi, "Experimental demonstration of segment routing," *J. Lightw. Technol.*, vol. PP, no. 99, early access on IEEE Xplore.

[13] A. Giorgetti, F. Cugini, F. Paolucci, and P. Castoldi, "OpenFlow and PCE architectures in wavelength switched optical networks," in *Proc. ONDM*, Apr. 2012.

[14] A. Sgambelluri, A. Giorgetti, F. Cugini, F. Paolucci, and P. Castoldi, "OpenFlow-based segment protection in Ethernet networks," *J. Opt. Commun. Netw.*, vol. 5, no. 9, pp. 1066–1075, Sep. 2013.

[15] A. Giorgetti, F. Paolucci, F. Cugini, and P. Castoldi, "Dynamic restoration with GMPLS and SDN control plane in elastic optical networks [invited]," *J. Opt. Commun. Netw.*, vol. 7, no. 2, pp. A174–A182, Feb. 2015.

[16] A. Giorgetti, F. Castoldi P., Cugini, J. Nijhof, F. Lazzeri, and G. Bruno, "Path encoding in segment routing," in *Proc. Globecom 2015*, Dec. 2015.

[17] "OpenFlow switch specification 1.3.2," https://www.opennetworking.org, Apr. 2013.

[18] A. Medina, A. Lakhina, I. Matta, and J. Byers, "BRITE: an approach to universal topology generation," in *Proc. MASCOTS*, 2001, pp. 346–353.

[19] A. Bukva, R. Casellas, R. Martinez, and R. Munoz, "A dynamic path-computation algorithm for a GMPLS-enabled multi-layer network," *J. Opt. Commun. Netw.*, vol. 4, no. 6, Jun. 2012.