Load-Dependent Flow Splitting for Traffic Engineering in Resilient OpenFlow Networks

Wolfgang Braun, Michael Menth
University of Tübingen, Department of Computer Science, Germany

***Abstract*—In this paper we propose and investigate load-dependent load balancing for resilient OpenFlow networks. The objective is to spare extra capacity for the primary path of a traffic aggregate (flow) by accommodating excess traffic on its backup path.**

The contribution of the paper is manyfold. We explain existing OpenFlow features for traffic monitoring and dynamic flow splitting, combine them to implement load-dependent balancing of a flow's traffic on its primary and backup paths, and propose three different load-dependent flow splitting (LDFS) policies. We develop a performance evaluation method to quantify the capacity for load-balancing and protection switching methods such that expected traffic can be accommodated for envisioned overload and failure scenarios. Finally, we assess the usefulness of LDFS by comparison with traffic-agnostic single- or multipath forwarding methods.

***Index Terms*—Software-defined networking, OpenFlow, traffic engineering, monitoring, load balancing**

I. Introduction

Network operation includes the possibility of network failures, traffic shifts, and traffic overloads. Network failures are considered typically short-lived [1] but protection schemes are necessary to minimize the traffic loss. Traffic overloads are often caused by inter-domain routing effects [2] or network failures. Re-configuration of the network can solve these problems. However, network updates can be difficult to perform properly [3], i.e., have to avoid micro-loops, or can have a notable impact on inter-domain routing [4]. A dynamic load balancing approach has the potential to handle temporary overloads. Then, the difficult reconfiguration process could only be applied for overloads that take place over longer time periods.

Software-defined networking (SDN) and OpenFlow [5] provide flexibility, programmability, and the ease of introduction of new services [6] in communication networks. The OpenFlow protocol [7] was continuously enhanced to support resilience, quality of service (QoS), and traffic engineering features. In this paper, we discuss the OpenFlow features that can be used to implement load-dependent flow splitting. This novel forwarding behavior can be used to handle temporary overloads on demand.

We propose *load-dependent flow splitting* (LDFS) that applies load balancing to a flow or aggregate based on the current load, i.e., traffic is normally sent over the primary path but traffic overloads trigger load balancing. We propose three policies how traffic can be distributed on multiple paths. Then, we provide a methodology to analyze routing methods with regard to traffic overload models and network scenarios. Finally, we analyze the performance impact of dynamic load balancing with LDFS compared to single-shortest path and multipath routing.

The remainder of this paper is structured as follows. Section II gives an overview of related work. In Section III, we describe the load-dependent flow splitting in detail as well as the implementation possibilities with OpenFlow. The evaluation methodology is described in Section IV. Section V provides the performance study and Section VI concludes this paper.

This work has been supported by the German Federal Ministry of Education and Research (BMBF) under support code 16BP12307 (EUREKA-Project SASER). The authors alone are responsible for the content of the paper.

II. Related Work

Traffic engineering (TE) for software-defined networking is discussed in [8]. They optimize the network with regard to network utilization, improved delay, and loss performance. They also cover the optimization process in networks where SDN is incrementally introduced.

Google describes in [9] their internal global network that interconnects their data centers worldwide and show how traffic engineering can be supported by routing decisions. Their traffic engineering server schedules operations depending on the available bandwidth in the network. They show significant effects in resource utilization in the context of data center synchronization and communication which may not be applicable to typical communication networks.

Multipath traffic engineering is considered in [10]. The authors propose the use of multiple paths from source to destination with optimized split ratios between those paths. A mixed-integer linear program (MILP) is defined that provides optimized path layouts with regard to state trade-offs. Self-protecting multipath [11] is a similar approach that uses multiple paths from source to destination to combine resilience and traffic engineering aspects. In contrast to our work, these methods operate leveraging results from an offline optimization process for configuration. We are investigating the impact of dynamic load balance dependent of the current rate on a flow.

Data plane resilience is an important aspect for our dynamic load balancing approach. Various works were presented that discuss practical aspects of protection schemes for OpenFlow networks. The authors of [12] are discussing the requirement

Group Identifier	Group Type	Counters	Action Buckets

Fig. 1: Group table entries for OpenFlow 1.1 and later.

of bidirectional forwarding detection (BFD) for OpenFlow switches to achieve fast protection. A BFD component is responsible to monitor the ports by sending *hello* and *echo* messages frequently over the links to detect network failures. The authors show the viability of the approach using experiments based on MPLS Transport Profile. In [13], the authors provide fast protection using BFD for the Open vSwitch and were able to show protection times between 3 ms and 30 ms in practice.

III. Load-Dependent Flow Splitting

In this section we explain load-dependent flow splitting (LDFS) for resilient OpenFlow networks. We discuss required OpenFlow features to implement our proposed routing mechanism. Finally, we describe three different load balancing options for the proposed method.

A. Requirements

OpenFlow networks consist of a separated control and data plane. The forwarding elements depend on the configuration of the controllers to operate properly. Therefore, we assume that backup paths are available to cover the interval of a disrupted switch-controller communication. Otherwise a switch cannot operate in a timely manner when a network failure in the control plane occurs. This is more problematic when the control and data plane share a common physical infrastructure because network failures affect both planes.

Typically, network operators provide additional backup capacities. We propose to use these capacities when traffic overload occurs, i.e., when a traffic flow or aggregate carries increased load, we perform load balancing on the primary and secondary path towards the destination. Otherwise, the switch sends traffic over the primary path unless a network failure happens. In such a case, the switch sends the traffic on the unaffected path.

Therefore, we require programmable load balancing for OpenFlow switches, be able to provide backup paths, and monitor the current rate of flows. In the following sections, we discuss these required features for OpenFlow with regard to the different specifications. Then, we provide a detailed description of LDFS and three different policies.

B. Load Balancing with OpenFlow Switches

Load balancing requires OpenFlow group tables which are defined in OpenFlow 1.1 and later. Group tables enable OpenFlow to perform additional forwarding methods such as flooding of packets, load balancing, fast failover, etc.

Multiple flow entries can point to a single group table entry and they are all forwarded in the same way. The structure of a group table entry is illustrated in Figure 1. It consists of an unique identifier, the group type, counters, and an ordered list of action buckets. An action bucket contains a set of actions that are executed on packets when the bucket is selected.

The group type defines the semantic of the group. The optional group type *select* provides load balancing by selecting one of the buckets for each packet. For example, if a group should load balance through port 2 and port 3, two action buckets have to be defined; the buckets contain the actions "forward to port 2" and "forward to port 3", respectively. The switch computes the action bucket to be chosen by a selection algorithm that is outside of the scope of the OpenFlow standard. Potential algorithms can be based on hash values over match fields or a simple round robin approach. Note that, the selection algorithm is only allowed to select *live* action buckets. We cover *liveness* in the next section.

C. OpenFlow Resilience

OpenFlow resilience is related to the control plane and to the data plane. Control plane resilience topics are resilient controller placement, multiple controller availability, etc. In this work, we require data plane resilience which is provided by backup paths. OpenFlow has optional support for backup paths through group tables which are described in Section III-B.

The group type *fast failover* implements backup paths. The actions of the first *live* action bucket are applied to packets. In the example above, packets are sent over port 2 until it goes down. Then, port 3 is used. Therefore, each action bucket requires a signaling mechanism to trigger the *liveness* of the group or the port. The OpenFlow specification leaves the implementation details open. Adding a bidirectional forwarding detection (BFD) component to an OpenFlow switch can achieve protection against failures within 50 ms [12], [13].

D. OpenFlow Monitoring Capabilities

The OpenFlow protocol defines counters for the main components of a switch. Counters exist for tables, ports, queues, and table entries. These counters are updated when packets are processed by the switch. Each flow table entry and each group table entry contain counters that measure the entry's duration, received packets, and received bytes. Although the received packet statistics are optional, they have the potential to provide the necessary statistics for flow and aggregate rates that are required to implement LDFS. However, statistic generation using counters is difficult to manage in practice due to hardware restrictions. This problem is described for OpenFlow switches in [14]. The authors provided a solution that is able to generate statistics more efficiently.

OpenFlow's monitoring capabilities were notably enhanced in OpenFlow 1.3. Meter tables enable OpenFlow to implement simple QoS operations. In combination with per-port queues, complex QoS frameworks such as DiffServ can be implemented. Meter table entries contain a list of meter bands. Each meter band has a specified rate on which it applies. The meter with the highest configured rate that is lower than the current measured rate is chosen. Thus, meters also have the potential to implement LDFS.

E. Load-Dependent Flow Splitting (LDFS) Policies

The load-dependent flow splitting algorithm operates on a set of individual flows or on macro flows. Macro flows can be defined using wildcard-based OpenFlow match rules. Individual flows can form an aggregate by applying the *group* instruction which causes those flows to be processed by the LDFS group. The rate of this aggregate is either measured using the group counters or by a meter that is attached to the corresponding flow table entries.

A new group type is required that behaves similarly to the *select* type. In addition, the group requires a threshold rate value T that distinguishes normal traffic from overload traffic. When the threshold rate is not exceeded, the first action bucket is executed. Otherwise, load balancing is performed on the action buckets. Only *live* action buckets can be selected.

We propose three load balancing variants which behave differently on overloads. In the following, we define the rate of an aggregate g as $r(g)$ and the excess traffic rate as $r_e(g) = r(g) - T$.

1) Balance All Traffic (BAT): When the threshold is exceeded we perform load balancing on the whole aggregate. Then, the *balance all traffic* (BAT) method has the same forwarding behavior as the multipath routing. Otherwise, the primary path is selected.

2) Balance Excess Traffic (BET): The *balance excess traffic* (BET) method is similar to the BAT method. However, only the excess traffic is split over available paths instead the whole aggregate traffic. This can be achieved by load balancing that aims to have a load of $T + \frac{r_e(g)}{2}$ on the primary and $\frac{r_e(g)}{2}$ on the secondary path.

3) Redirect Excess Traffic (RET): The *redirect excess traffic* (RET) method redirects all excess traffic on the backup path. A load balancing algorithm that sends a rate of T on the primary and a rate of $r_e(g)$ on the secondary path is a potential implementation of RET.

IV. Methodology

In this section, we describe and explain the considered traffic models. We present the network scenarios that model the combination of network failures and traffic overloads and discuss the computation of required link capacities depending on a routing mechanism. Finally, we discuss the path computation for multipath routing and LDFS.

A. Notation

A network is a graph $G = (V, E)$ where V is the set of nodes and E is the set of edges or links. We denote the set of aggregates as $\mathcal{G}$. Each aggregate $g \in \mathcal{G}$ is characterized by its source and destination ($g = (s, d)$) and has a rate $r(g)$.

B. Traffic Models

We generate traffic matrices with the gravity model [15] for the performance analysis. The gravity model uses a population function $\pi(n)$ on nodes $n \in V$ and distributes the overall traffic C_{tot} in such a way that more populated nodes send and receive more traffic than less populated ones. The aggregate rates $r(g)$ are generated with

$$r(g = (v, w)) = \begin{cases} \frac{\pi(v) \cdot \pi(w) \cdot C_{tot}}{\sum_{x,y \in V, x \neq y} \pi(x) \cdot \pi(y)} & \text{if } s \neq d \\ 0 & \text{if } s = d \end{cases}$$

We investigate three different overload traffic models.

1) Hot Spot Model: The hot spot model [16] is a conservative overload model that shifts traffic locally to hot spots. The overall traffic C_{tot} in the network is constant, i.e., flows attached to a hot spot have increased load while other flows experience reduced load. This model is implemented by applying an overload factor f_o to the population function

$$\pi^v_{hotspot}(w) = \begin{cases} \pi(w) & \text{if } v \neq w \\ f_o \cdot \pi(w) & \text{if } v = w \end{cases}$$

2) Node Overload Model: The node overload model increases the overall network rates which can be caused by inter-domain routing [16], [2]. An overloaded node causes all attached flows to carry a multiple of the original load, i.e., the rate is multiplied with the overload factor f_o.

3) Aggregate Overload Model: The aggregate overload model shares the same basic idea as the node overload model. In contrast, an aggregate instead of a node can be overloaded having a rate multiplied with the overload factor f_o.

C. Network Scenarios

We consider network failures and overloads. We model network failures with link failures and traffic overloads with the models described in Section IV-B. We refer to the combination of link failures and traffic overloads as network scenario. The scenario specifies all failed links and the traffic matrix containing the rates of all aggregates.

We define various sets of network scenarios $S^{i,j}$. This set contains all scenarios that consist up to i simultaneous link failures in combination with up to j simultaneous overloads. For example, $S^{0,0}$ contains one scenario: the failure-free case with no overloads, $S^{1,0}$ contains all single link failures without overloads, etc.

We compute the required capacity for a routing mechanism and a set of network scenarios S by measuring the traffic rate for the routing mechanism with regard to the link failures and the traffic matrix. We compute the required link capacity $c(l, s)$ for each link in each scenario $s \in S$. The required capacity for a link is $c(l) = \max_{s \in S} c(l, s)$.

D. Path Computation for Multipath and LDFS

We compare single-shortest path routing, multipath routing (MPR), and the LDFS variants (BAT, BET, RET). MPR and LDFS both use the 2-shortest paths to highlight the impact of the dynamic load balancing approach.

For each source and destination, two shortest paths are computed that are maximally disjoint, i.e., share the least number of links. The paths are computed with a k-disjoint shortest path algorithm [17]. The shorter path is the primary and the longer the secondary path. Note that this primary path can be longer than the shortest path from the source to the destination.

V. Results

In this section, we present the performance evaluation of our proposed method. First, we show the networks under study and explain our metrics. We present our results and discuss the impact of load-dependent flow splitting compared to single-shortest path and multipath routing.

A. Networks under Study

We provide results for the Agis, Abilene, and GÉANT networks that are publicly available at the "topology zoo" [18]. The Agis and Abilene networks are long-haul backbone networks in the USA. The former one consists of 25 nodes and 30 bidirectional links and the latter one consists of 11 nodes and 14 links. The GÉANT network is a European research network with 40 nodes and 61 links. We selected the human population to generate the traffic matrices for the population function [19], [20].

B. Metrics

We denote required link capacities for a set of network scenarios S and a routing method m as $c(S, m, l)$. The computation of these capacities is described in Section IV-C.

The required network capacity is the sum of all link capacities. The maximum link capacity can be an interesting metric because higher rate interfaces are generally more expensive than lower rate interfaces.

We use values relative to single-shortest path routing (SPR) to simplify the comparison of multipath routing and LDFS. Thus, we define the relative required network capacity as

$$C_{net}(S, m) = \frac{\sum_{l \in E} c(S, m, l)}{\sum_{l \in E} c(S, \mathrm{SPR}, l)}$$

and the relative maximum link capacity as

$$C^{max}_{link}(S, m) = \frac{\max_{l \in E} c(S, m, l)}{\max_{l \in E} c(S, \mathrm{SPR}, l)}$$

C. Impact on Required Network Capacity

We present the results for the required network capacity for the GÉANT network, the hot spot model, and an overload factor of $f_o = 3$. The results of the other networks, models, and overload factors are very similar and additional figures do not provide further insight. However, differences are explicitly stated in the text.

Figure 2 shows the relative required network capacity in the GÉANT network for various network scenarios. The failure-free case without overloads ($S^{0,0}$) leads to approximately the same required network capacity for LDFS and SPR because LDFS only uses the primary paths. MPR requires additional 20% capacity which is caused by the usage of longer paths. In single link failure scenarios ($S^{1,0}$), both MPR and LDFS results in 8% less capacity compared to SPR.

In overload scenarios without failures ($S^{0,1}, S^{0,2}$), BET leads to less required network capacity than SPR and significantly less than MPR, BAT, and RET. In the Agis network, we reduced the required network capacity by approximately 13% for BET compared to SPR. RET performs worse than the other

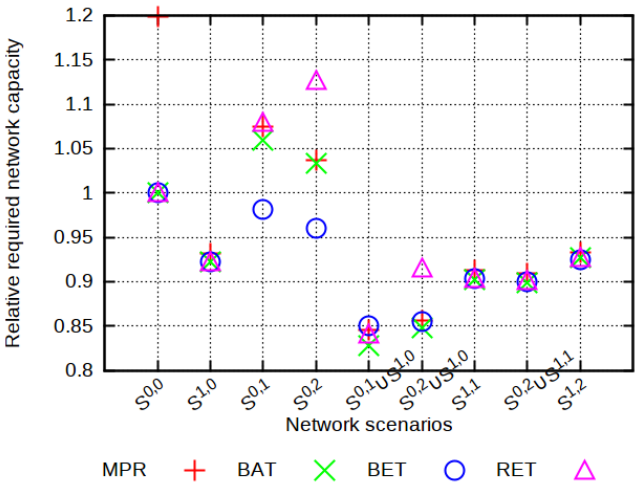

Fig. 2: The relative required network capacity in the GÉANT network for various network scenarios. The traffic overload is modeled by hot spots with a factor of 3.

routing methods. When the overload factor is increased, the required network capacity for RET increases significantly. This is caused by sending a lot of traffic over the longer secondary path when the load significantly exceeds the threshold rate.

For non-simultaneous overloads and failures scenarios ($S^{0,1} \cup S^{1,0}, S^{0,1} \cup S^{1,0}$) we observe additional savings of 10% compared to SPR. Yet, MPR and LDFS do not differ so clearly in the GÉANT network. The BET method performs slightly better than the other mechanisms in the Agis and Abilene networks.

Almost no differences can be observed for the different routing methods in network scenarios with simultaneous link failures and traffic overloads ($S^{1,1}, S^{2,0} \cup S^{1,1}, S^{1,2}$). In these network scenarios, it is possible that one of the paths of an overloaded flow is affected by a link failure. Then, there is no path choice for both MPR and LDFS. Note that the RET method provides notably higher required network capacities for high overload factors than the other methods in these network scenarios.

D. Impact on Maximum Link Capacity

Figure 3 shows the relative maximum link capacity in the GÉANT network for various network scenarios. In $S^{0,0}$, LDFS methods are close to SPR and MPR leads to almost 30% less maximum link capacity. However, for the Agis network we observe significantly higher maximum link capacities for the LDFS methods. Surprisingly, MPR performs worse in the Abilene network which may be caused by the topology.

The maximum link capacities are equal for the routing methods in network scenarios with failures ($S^{1,0}, S^{1,1}, S^{2,0} \cup S^{1,1}, S^{1,2}$). We again explain this fact by the lack of choice for MPR and LDFS in certain failure scenarios with simultaneous link failures and overloads. In the GÉANT network, the maximum link load is between 5–10% lower compared to SPR. In the Agis and Abilene network it is close to SPR.

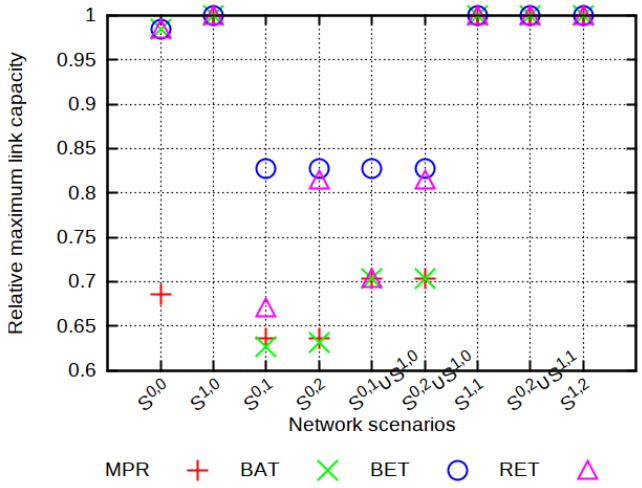

Fig. 3: The relative maximum link capacity in the GÉANT network for various network scenarios. The traffic overload is modeled by hot spots with a factor of 3.

The maximum link capacities differ significantly in network scenarios with non-simultaneous link failures and overloads. MPR and BAT often perform quite similar and requires up to 37% less capacity compared to SPR. MPR and BAT behave equal in overload scenarios which explains the similarity with regard to maximum link capacities. The BET method performs better than SPR but approximately 12–20% worse than MPR. In the Agis and Abilene networks, MPR and LDFS only perform better than SPR in $S^{0,1}$ and $S^{0,2}$. In addition, the maximum link capacity for BET is similar to MPR in the Abilene network.

VI. Conclusion

In this work we proposed how load-dependent flow-splitting (LDFS) can be implemented using monitoring and fast failover mechanisms of OpenFlow, and suggested three different load balancing policies (BAT, BET, and RET) for traffic engineering. We compared the three LDFS variants with traffic-agnostic single-shortest-path and 2-shortest-paths routing (SPR, MPR) with regard to required network capacity and maximum link capacity. A special LDFS requires least capacity when networks are expected to accommodate traffic for non-simultaneous link failures and traffic overloads, but MPR requires least link capacities. However, when networks were provisioned for simultaneous failures and overloads, capacity savings by LDFS diminished.

Acknowledgements

The authors thank Christian Duta for his programming efforts.

References

[1] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of Link Failures in an IP Backbone," in *ACM SIGCOMM Internet Measurement Workshop*, Marseille, France, Nov. 2002, pp. 237 – 242.

[2] T. Schwabe and C. G. Gruber, "Traffic Variations Caused by Inter-domain Re-routing," in *International Workshop on the Design of Reliable Communication Networks (DRCN)*, Ischia Island, Italy, Oct. 2005.

[3] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, "Abstractions for Network Update," in *ACM SIGCOMM*, 2012, pp. 323–334.

[4] L. Vanbever, S. Vissicchio, L. Cittadini, and O. Bonaventure, "When the Cure is Worse than the Disease: the Impact of Graceful IGP Operations on BGP," in *IEEE Infocom*, 2013.

[5] 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 Computer Communications Review*, vol. 38, no. 2, pp. 69–74, 2008.

[6] W. Braun and M. Menth, "Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices," *Future Internet*, vol. 6, no. 2, pp. 302–336, 2014. [Online]. Available: http://www.mdpi.com/1999-5903/6/2/302

[7] OpenFlow Switch Consortium and others, "OpenFlow Switch Specification Version 1.4.0," Oct. 2013. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.4.0.pdf

[8] S. Agarwal, M. Kodialam, and T. Lakshman, "Traffic Engineering in Software Defined Networks," in *IEEE Infocom*, April 2013, pp. 2211–2219.

[9] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat, "B4: Experience with a Globally-Deployed Software Defined WAN," in *ACM SIGCOMM*, Aug. 2013.

[10] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers," in *ACM SIGCOMM*, Barcelona, Spain, Aug. 2009.

[11] M. Menth, R. Martin, and U. Spörlein, "Optimization of the Self-Protecting Multipath for Deployment," in *in Legacy Networks," in IEEE International Conference on Communications (ICC)*, 2007.

[12] J. Kempf, E. Bellagamba, A. Kern, D. Jocha, A. Takàcs, and P. Sköldström, "Scalable Fault Management for OpenFlow," in *IEEE International Conference on Communications (ICC)*, 2012, pp. 6606–6610.

[13] N. L. M. van Adrichem, B. J. van Asten, and F. A. Kuipers, "Fast Recovery in Software-Defined Networks," in *European Workshop on Software Defined Networks (EWSDN)*, 2014.

[14] J. C. Mogul and P. Congdon, "Hey, You Darned Counters!: Get off my ASIC!" in *ACM Workshop on Hot Topics in Software Defined Networks (HotSDN)*, 2012, pp. 25–30.

[15] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot, "Traffic Matrix Estimation: Existing Techniques and New Directions," in *ACM SIGCOMM*, Pittsburgh, USA, Aug. 2002.

[16] M. Menth, M. Duelli, R. Martin, and J. Milbrandt, "Resilience Analysis of Packet-Switched Communication Networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 6, pp. 1950–1963, Dec 2009.

[17] J. W. Suurballe and R. E. Tarjan, "A Quick Method for Finding Shortest Pairs of Disjoint Paths," *Networks Magazine*, vol. 14, pp. 325–336, 1984.

[18] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765 –1775, Oct. 2011.

[19] Population Reference Bureau, "2012 World Population Data Sheet," Oct. 2014. [Online]. Available: http://www.prb.org/pdf12/2012-population-data-sheet_eng.pdf

[20] United States Census Bureau, "Population Estimates," Oct. 2014. [Online]. Available: http://www.census.gov/popest/data/cities/totals/2013/SUB-EST2013.html