FACT: Flow-based Approach for Connectivity Tracking

Dominik Schatzmann[1], Simon Leinen[2], Jochen Kögel[3], and Wolfgang Mühlbauer[1]

[1] ETH Zurich, {schatzmann,muehlbauer}@tik.ee.ethz.ch
[2] SWITCH, simon.leinen@switch.ch
[3] University of Stuttgart, jochen.koegel@ikr.uni-stuttgart.de

Abstract. More than 20 years after the launch of the public Internet, operator forums are still full of reports about temporary unreachability of complete networks. We propose FACT, a system that helps network operators to track connectivity problems with *remote* autonomous systems, networks, and hosts. In contrast to existing solutions, our approach relies solely on *flow-level* information about observed traffic, is capable of *online data processing*, and is *highly efficient* in alerting only about those events that *actually affect* the studied network or its users.
We evaluate FACT based on flow-level traces from a medium-sized ISP. Studying a time period of one week in September 2010, we explain the key principles behind our approach. Ultimately, these can be leveraged to detect connectivity problems and to summarize suspicious events for manual inspection by the network operator. In addition, when replaying archived traces from the past, FACT reliably recognizes reported connectivity problems that were relevant for the studied network.

Keywords: monitoring, connectivity problems, flow-based

1 Introduction

"Please try to reach my network 194.9.82.0/24 from your networks ... Kindly anyone assist", (NANOG mailing list [1], March 2008). Such e-mails manifest the need of tools that allow to monitor and troubleshoot connectivity and performance problems in the Internet. This particularly holds from the perspective of an individual network and its operators who want to be alerted about disrupted peerings or congested paths before customers complain.

Both researchers [2,3,4,5] and industrial vendors [6,7] have made proposals for detecting and troubleshooting events such as loss of reachability or performance degradation for traffic that they exchange with other external networks, unfortunately with mixed success. Predominantly, such tools rely on active measurements using ping, traceroute, etc. [2,4]. Besides, researchers have suggested to leverage control plane information such as publicly available BGP feeds [8,3,9], although Bush et al. [10] point out the dangers of relying on control-plane information. Other concerns about existing tools include a high "dark" number of undetected events [8], a narrow evaluation solely in the context of a testbed or small system [9,5], or the time gap between the occurrence of an event and its observation and detection [8].

In this paper we propose FACT, a system that implements a **F**low-based **A**pproach for **C**onnectivity **T**racking. It helps network operators to monitor connectivity with *remote* autonomous systems (ASes), subnets, and hosts. Our approach relies on *flow-level* information about observed traffic (and not on control-plane data), is capable of *online data processing*, and *highly efficient* in alerting only about those events that *actually affect* the monitored network or its users.

In contrast to existing commercial solutions [7,6], we do not consider aggregate traffic volumes per interface or per peering to detect abnormal events, but pinpoint on a *per-flow basis* those cases where external hosts are unresponsive. On the one hand, this requires careful data processing to correctly handle asymmetric routing and to eliminate the impact of noise due to scanning, broken servers, late TCP resets, etc. On the other hand, our flow-based approach allows to compile accurate lists of unresponsive network addresses, which is a requirement for efficient troubleshooting.

To test our system we rely on a one-week flow-level trace from the border routers of a medium-sized ISP [11]. We demonstrate that our approach can be leveraged to detect serious connectivity problems and to summarize suspicious events for manual inspection by the network operator. Importantly, replaying flow traces from the past, FACT also reliably recognizes reported connectivity problems, but only if those are relevant from the perspective of the studied network and its users. Overall, we believe that our approach can be generally applied to small- to medium-sized ISPs, and enterprise networks. In particular networks that (partially) rely on default routes to reach the Internet can strongly benefit from our techniques, since they allow to identify critical events even if these are not visible in the control plane information.

2 Methodology

Our goal is to enable network operators to monitor whether remote hosts and networks are reachable from inside their networks or their customer networks, and to alert about existing connectivity problems. Such issues include cases where either we observe a significant number of unsuccessful connection attempts from inside the studied network(s) to a specific popular remote host, or where many remote hosts within external networks are unresponsive to connection attempts originated by potentially different internal hosts.

To obtain a network-centric view of connectivity, we rely on flow-level data exported by *all* border routers of a network, see Fig. 1. In this regard, our approach is generally applicable to all small- and medium-sized ISPs, and enterprise networks. Monitoring the complete unsampled traffic that crosses the border of our network allows to match outgoing with incoming flows and to check for abnormal changes in the balance between incoming and outgoing flows for external

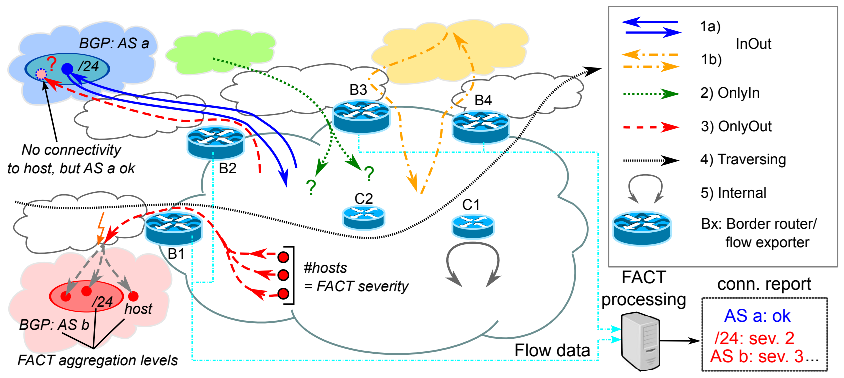


Fig. 1. Measurement infrastructure and flow types.

endpoints at different *aggregation levels* (hosts or networks). In particular networks that (partially) rely on default routes to reach the Internet can strongly benefit from such an approach, since it allows to identify critical events even if these are not visible in the control plane information.

As shown in Fig. 1, we distinguish between five *flow types*: `Internal` connections never cross the network border, and thus are neither recorded nor studied further in our approach. Since the scope of this paper is limited to cases where remote hosts or networks are unresponsive to connection attempts originated by internal hosts, we ignore flows that traverse our network (`Traversing`) or flows for which we cannot find traffic in the outbound direction (`OnlyIn`), e.g., caused by inbound scanning. If we can associate outgoing flows with incoming flows, we assume that external hosts are reachable (`InOut`) and also take this as a hint that there exists connectivity towards the remote network. Note that the incoming flow can enter the network via the same border router that was used by the outgoing flow to exit the network. Yet, due to the asymmetric nature of Internet paths this is not necessary [9]. Finally, we observe flows that exit the network but we fail to find a corresponding incoming response (`OnlyOut`).

To detect potential connectivity problems, we focus on the latter category `OnlyOut`. Note that we rely on the assumption that our measured flow data is complete, i.e., for any outgoing flow the associated incoming flow is observed by our collection infrastructure provided that there has been a response in reality. Evidently, network operators only want to get informed about critical events that include loss of connectivity towards complete networks or towards popular hosts that a significant share of internal hosts tries to reach. Our approach to achieve this goal is twofold.

First, we heavily rely on data *aggregation* to investigate connectivity towards complete networks. More precisely, we aggregate occurrences of `OnlyOut` flow types across external hosts, /24 networks, or prefixes as observed in public BGP routing tables. For example, only if we observe within a certain time period a considerable number of `OnlyOut` flow types towards different hosts of a specific

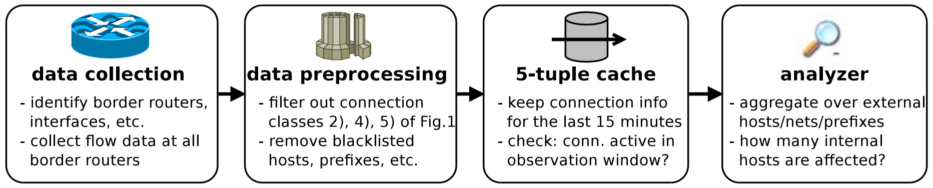

data collection
- identify border routers, interfaces, etc.
- collect flow data at all border routers
data preprocessing
- filter out connection classes 2), 4), 5) of Fig.1
- remove blacklisted hosts, prefixes, etc.
5-tuple cache
- keep connection info for the last 15 minutes
- check: conn. active in observation window?
analyzer
- aggregate over external hosts/nets/prefixes
- how many internal hosts are affected?

Fig. 2. Architectural components of FACT.

external network, and no `InOut` types, we conclude that the complete external network is currently not reachable for internal hosts. Hence, our decision is not based on observed connectivity between a single pair of internal and external hosts.

Second, we take into account the number of internal hosts that are affected by connectivity problems towards a host, network, or BGP prefix, i.e., the *severity of an observed event*. For example, loss of connectivity towards an individual external host is interesting for a network operator if a large number of different internal hosts fail to reach such a popular service. Moreover, knowing the number of affected internal hosts is crucial to extract short summaries of candidate events which network operators can check manually in reasonable time.

3 Data Sets

We investigate our approach based on data collected in the SWITCH network [11], a medium-sized ISP in Switzerland connecting approximately 30 Swiss universities, government institutions, and research labs to the Internet. The IP address range contains about 2.2 million internal IP addresses. For our studies we have collected a trace in September 2010 (`OneWeek`) that spans 7 days and contains unsampled NetFlows summarizing all traffic crossing the 6 border routers of the SWITCH network. This results in $14 - 40k$ NetFlow records per second. In addition to `OneWeek` we extract some shorter traces to study selected connectivity problems from the past, see Section 5.

4 Connectivity Analysis

The implementation of FACT includes four major components, see Fig. 2. After `data collection`, a `preprocessing` step removes some flows from the data stream, e.g., blacklisted hosts or information that is not needed to achieve our goals. For a limited time we keep the remaining flows in the `5-tuple cache`, which is continuously updated with the latest flow information. In the following we will provide more details about the implementation of the individual components.

4.1 Data Collection and Preprocessing

In addition to standard flow information including IP addresses, port numbers, protocol number, packet counts, byte counts, etc., we store identifiers for the border routers and interfaces over which traffic with external networks is exchanged. Next, we exclude a considerable number of unnecessary flows to save memory and computational resources, but also eliminate flows that have turned out to be harmful for the detection of connectivity problems. Such flows include for example traffic from/to PlanetLab hosts or bogon IP addresses, and multicast. For now, we generate an appropriate blacklist manually, but we plan to automate this process in the future. For reasons already described in the preceding section, we remove in this step also all flows of the class `Traversing` and `Internal`, see Fig. 1.

4.2 5-tuple cache

The subsequent data processing respects the fact that the active timeout of our flow collection infrastructure is set to 5 minutes.[4] Therefore, we partition the timeline into intervals of 5 minutes and proceed with our data processing whenever such a time interval has expired. Our goal is to maintain for each interval a hash-like data structure (*5-tuple cache*) that, for observed flows identified by IP addresses, protocol number, and application ports, stores and updates information that is relevant for further analysis. This includes packet counts, byte counts, information about the used border router and the time when the flows were active for the in and out flow. Note that at this point we implicitly merge unidirectional to bidirectional flows (biflows).

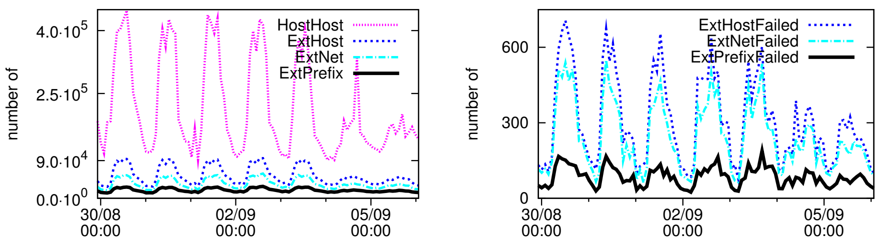
After the time interval has expired we extract from the obtained biflows and remaining unidirectional flows two sets: The set `ConnSuccess` includes those biflows of type *InOut* where at least one of the underlying unidirectional flows starts or ends within the currently studied time interval and are initiated by internal hosts[5]. The second set, called `ConnFailed`, includes only those unidirectional flows of type `OnlyOut` where the outgoing flow either starts or ends in the currently studied time interval. To reduce the effect of delayed packets (e.g., TCP resets), we here ignore unidirectional flows if a corresponding reverse flow has been observed during any of the preceding time intervals.[6] All other flows of the *5-tuple cache* that are not in the set `ConnSuccess` or `ConnFailed` are excluded from further consideration for this time interval.

While `ConnSuccess` flows indicate that an internal host in our network can indeed reach the external host, we take occurrences of `ConnFailed` as a hint for potential connectivity problems with the remote host. However, the latter assumption does not necessarily hold when applications (e.g., NTP or multicast)

[4] After 5 minutes even still active flows are exported to our central flow repository.

[5] We rely on port numbers to determine who initiates a biflow.

[6] Our hash-like data structure is not deleted after a time period of 5 minutes but continuously updated. Only if a biflow is inactive for more than 900 seconds, it is removed from our hash-like data structure.



(a) Visible external destinations. (b) Unresponsive external destinations.

Fig. 3. External hosts, networks, and prefixes.

are inherently unidirectional. Hence, we exclusively take into account HTTP traffic using port 80, which is symmetric by nature and due to its popularity visible in any type of network.[7] More marginal fine-tuning of our data processing is required. Yet, given space limitations we refrain from providing more details.

4.3 Analyzer

To study observed connectivity with remote hosts and to detect potential problems, the analyzer component processes the sets `ConnSuccess` and `ConnFailed` every 5 minutes. We aggregate `ConnFailed` and `ConnSuccess` flows for the same pair of internal and external host if we find more than one flow, possibly with different port numbers. The obtained host-host tuples are classified as `HostHostSuccess` if at least one `ConnSuccess` flow has been identified, `HostHostFailed` otherwise. Based on this initial aggregation step, we independently compute three stronger aggregation levels: we group host-host tuples into one tuple if they affect the same external host (`ExtHostSuccess` or `ExtHostFailed`), the same external /24 network (`ExtNetSuccess` or `ExtNetFailed`), and BGP prefixes (`ExtPrefixSuccess` or `ExtPrefixFailed`). With respect to the last granularity, we use publicly available BGP routing tables to determine the corresponding BGP prefix for a given external host. Again, we classify an aggregate class as `Success` if at least one tuple is marked as `HostHostSuccess`.

Fig. 3 displays the number of visible and unresponsive external destinations if the three aggregation granularities are applied to `OneWeek`, see Section 3. According to Fig. 3(a) the absolute number of visible external destinations shows a strong daily and weekly pattern irrespective of the used aggregation level. Aggregating from host-host into `ExtHostFailed` and `ExtHostSuccess`, respectively, reduces the peaks from $525K$ to $90K$ tuples (/24s: $50K$, prefixes: $25K$). This provides evidence for the high visibility that our data has on external networks. However, Fig. 3(b) reveals that generally only a small fraction of external hosts (peaks of 700) are unresponsive and therefore classified as `ExtHostFailed` according to

[7]Experiments relying on DNS traffic turned out to work as well.

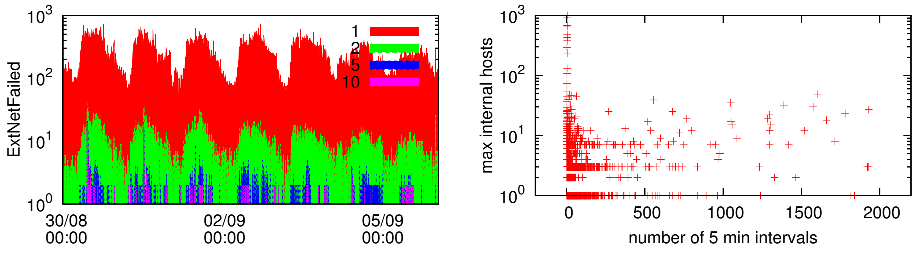
our methodology. This fraction is significantly smaller for ExtNetFailed (peaks of 600) and ExtPrefixFailed (peaks of 180), respectively.

However, to cope with daily and weekly fluctuations and to limit the degree to which a single internal host (e.g., a scanning host) can impact our connectivity analysis, we need to take into account the *severity of an observed event* as well. By this we understand the number of internal users that actually fail to establish connectivity with a specific external host, /24 network, or BGP prefix during our 5 minute time intervals. Figure 4(a) displays the number of external /24 networks that are unresponsive to 1, 2, 5, and 10 internal hosts for the time spanned by OneWeek. The majority of these ExtHostFailed "events", namely 98%, only affect 1 internal host.

Yet, here it is important to study Fig. 4(b). It is also based on OneWeek and counts for every external host the number of 5-minute time intervals for which it has been classified as ExtHostFailed. This number (x-axis) is plotted against the maximum number of internal hosts (y-axis) that failed to establish connectivity with this external host (ExtHostFailed) at *any* 5-minute interval of OneWeek. We find that the majority of external hosts (96%) are only unresponsive in less than 10 time intervals of our trace. However, some hosts are unresponsive most of the time, e.g., abandoned ad servers. Data preprocessing as described in Section 4.1 could be refined to automatically blacklist such hosts and possibly their networks. Finally, we observe few external hosts that are unresponsive only during a small number of time intervals, but with a high maximum number of affected internal hosts. Cross-checking with technical forums in the Internet, we find that these events include for example a Facebook outage on August 31, 2010.

We point out that the data processing in FACT is faster than real time for SWITCH, a medium-sized ISP covering an estimated 6% of the Internet traffic in Switzerland and approximately 2.2 million IP addresses: flow data spanning 5 minutes[8] can be processed using a single thread in less than three minutes with a maximum memory consumption of less than 4GB. Aging mechanisms for our data structures[6] ensure that the overall memory consumption does not increase during long-term use of our system. Due to hash-like data structures we can access individual flows in our 5-tuple cache in constant time. The total time required for data processing mainly depends on the number of active flows. In principle, it is even possible to parallelize our processing by distributing the reachability analysis for different external networks to different CPU cores or physical machines. Yet, we leave it to future work to study FACT's performance for large tier-1 ISPs and how to make it robust against potentially higher false positive rates if sampled flow data is used.

[8]We see up to 200 million flows per hour.



(a) Number of `ExtNetFailed` that are unresponsive to 1, 2, 5, and 10 internal hosts.

(b) Frequency analysis of `ExtHostFailed` events

Fig. 4. Severity of observed events.

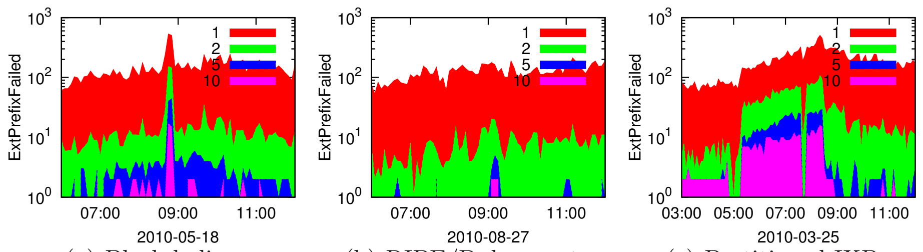
5 Case Studies

In this section we present a short analysis of three connectivity problems that were either detected by the network operator or publicly documented. To analyze those cases, we rely on data collected as discussed in Section 3.

Black-holing: On May 18, 2010, all services in an external /24 network were not accessible from SWITCH between 08:30 and 08:45. According to the operators of SWITCH, this problem was most likely due to a tier-1 provider that black-holed parts of the reverse traffic towards SWITCH. Yet, at this time the operators could only speculate how many hosts and customers, or even other /24 networks were affected by this problem. Applying FACT we confirm that the reported /24 network is indeed reported as unreachable at around 08:30. Surprisingly, FACT reveals that the overall number of unreachable hosts and /24 networks has doubled compared to the time before 08:30 while the number of unresponsive BGP prefixes is increased by a factor of even 6, see Fig. 5(a). Moreover, the reported /24 network is not even in the top ten list of the most popular unresponsive networks. This suggests that the impact of this event has been more serious than previously believed.

RIPE/Duke event: On August 27, 2010, some parts of the Internet became disconnected for some 30 minutes due to an experiment with new BGP attributes by RIPE and Duke University [12]. FACT reveals that at around 08:45 the number of popular unresponsive /24 networks indeed doubled. According to Fig. 5(b), for some BGP prefixes more than 15 internal hosts failed to establish connectivity. Yet, overall our analysis reveals that the impact of this incident on SWITCH and its customers was quite limited compared to the public attention that this event obtained.

Partitioned IXP: After scheduled maintenance by AMS-IX, SWITCH's connection to that exchange point came back with only partial connectivity. Some next-hops learned via the route servers weren't reachable, creating black holes. The next morning, several customers complained about external services being unreachable. Overall, it took more than four hours until the problem was finally solved by resetting a port. Fig. 5(c) shows that the number of unresponsive BGP



(a) Black-holing (b) RIPE/Duke event (c) Partitioned IXP

Fig. 5. Case studies: unresponsive BGP prefixes

prefixes is almost ten times higher than normal, over a time period of more than four hours. We believe that FACT would have helped to detect such a serious problem much faster and provided valuable hints about the origin of the problem.

6 Related Work

Approaches for detecting and troubleshooting reachability problems can be generally classified into two classes: *active probing* and *control plane based*.

With respect to *active probing*, Paxson et al. [9] are probably the pioneers to use traceroute for studying end-to-end connectivity between a (limited) set of Internet sites. Zhang et al. [2] perform collaborative probing launched from Planetlab hosts to diagnose routing event failures. Commercial solutions such as NetQoS [7] or Peakflow [6] generally rely on *active* measurements using ping, traceroutes, or continuous SNMP queries to network devices. Moreover, they frequently aggregate traffic volumes per interface, peering links, etc. to detect abnormal events, and hence do not base their analysis on a flow-level granularity as our work suggests. In contrast to active probing, the passive monitoring approach of FACT does not impose any traffic overhead and, importantly, only creates alerts for those unreachable hosts/networks that users actually want to access. Finally, FACT avoids an intrinsic problem of active probing techniques such as ping or traceroute, namely the implicit assumption that reachable hosts actually do respond to such tools.

In addition to active probing, a considerable number of research papers, e.g., [8,13] rely almost exclusively on *control-plane information* in the form of BGP routing feeds. However, Bush et al. [10] have clearly pointed out the dangers of such an approach, e.g., the wide-spread existence of default routes. In contrast, FACT is able to detect unreachability at multiple and finer granularities (e.g., on a host basis) than any approach that is purely based on routing data.

Later work including e.g., Hubble [3] and iPlane [4] rely on hybrid approaches combining active measurements with BGP routing information. Feamster et al. [14] adopt such an approach to measure the effects of Internet path faults on reactive routing. Overall, we believe that the passive approach adopted by FACT is very powerful compared to active probing and control-plane based techniques.

Yet, we plan to integrate active probing into our system to crosscheck detected reachability problems and to pinpoint the underlying causes.

7 Conclusion

We have proposed FACT, an online data processing system that helps operators to acquire facts about connectivity problems with remote autonomous systems, subnets, and hosts. In contrast to existing solutions, our approach relies solely on flow-level information extracted from traffic crossing the border of the network. We showed, with the help of reported real-world events, that FACT can be used to alert only about those events that actually affect the studied network or its users. Importantly, data processing of FACT is already faster than real time for a medium-sized ISP.

In the future we plan to refine and integrate our techniques into existing tracing tools (e.g., nfdump), to generate alerts based on automatically determined thresholds, and to provide summary reports that allow network operators to quickly troubleshoot connectivity problems. Ultimately, we plan to make our implementation of FACT available for public use.

References

1. Nanog mailing list. website. `http://www.nanog.org/mailinglist/`.
2. Y. Zhang, M. Mao, and M. Zhang. Effective diagnosis of routing disruptions from end systems. In *Proc. NSDI*, 2008.
3. E. Katz-Bassett, H. Madhyastha, J. John, A. Krishnamurthy, D. Wetherall, and T. Anderson. Studying black holes in the Internet with Hubble. In *Proc. NSDI*, 2008.
4. H. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: an information plane for distributed services. In *Proc. OSDI*, 2006.
5. M. Zhang, C. Zhang, V. Pai, L. Peterson, and R. Wang. PlanetSeer: Internet path failure monitoring and characterization in wide-area services. In *Proc. OSDI*, 2004.
6. Arbor Networks - Peakflow. website. `www.arbornetworks.com`.
7. CA technologies - NetQoS performance center. website. `www.netperformance.com/`.
8. J.Wu, M. Mao, J. Rexford, and J. Wang. Finding a needle in a haystack: Pinpointing significant BGP routing changes in an IP network.
9. V. Paxson. End-to-end routing behavior in the Internet. *IEEE/ACM Trans. Networking*, 5(5), 1997.
10. R. Bush, O. Maennel, M. Roughan, and S. Uhlig. Internet optometry: assessing the broken glasses in Internet reachability. In *Proc. of ACM IMC*, 2009.
11. The Swiss Education and Research Network (SWITCH). `http://www.switch.ch`.
12. RIPE/Duke event. `http://labs.ripe.net/Members/erik/ripe-ncc-and-duke-university-bgp-experiment/`.
13. A. Feldmann, O. Maennel, M. Mao, A. Berger, and B. Maggs. Locating Internet routing instabilities. *Proc. of ACM SIGCOMM*, 2004.
14. N. Feamster, D. Anderson, H. Balakrishnan, and F. Kaashoek. Measuring the effects of Internet path faults on reactive routing. In *Proc. of ACM SIGMETRICS*, 2003.