MeTRO: Low Latency Network Paths with Routers-on-Demand

Marc X. Makkes[1,2], Ana-Maria Oprescu[1], Rudolf Strijkers[1,2], Cees de Laat[1], and Robert Meijer[1,2]

[1] University of Amsterdam, The Netherlands
[2] TNO Information and Communication Technology, The Netherlands
{marc.makkes,rudolf.strijkers,robert.meijer}@tno.nl,
{a.m.oprescu,delaat}@uva.nl

Abstract. The current Internet is a loose federation of independent providers (ISPs) that manually manage inter-domain (ASes) route policies to primarily serve their own interests. The end-user experience may be hindered by two aspects: the ASes only optimize locally, possibly delivering sub-optimal end-to-end connections; the manual management of routing policies for a large amount of prefixes is error-prone. Infrastructure as a Service (IaaS) clouds let users allocate compute resources on demand, at different geographical locations, while Internet connectivity is guaranteed. Therefore, cloud providers represent untapped resources for a better end-user (application) Internet connectivity experience.

In this work we present MeTRO, a framework to construct better than best-effort routed Internet paths. Our method exploits the fact that cloud computer resources may host virtual routers and that one such router can be part of a path between two end systems. We perform an extensive evaluation of our method, by deploying it over 75 NLNOG Ring hosts. We show that our method, practically acting as an overlay network, decreases the latency in 58% of the cases studied, albeit increasing the number of hops. Our framework is specifically useful for monitoring and debugging failures, as well as configuration errors related to Internet reachability.

Keywords: Control framework, cloud computing, programmable networks, virtual routers.

1 Introduction

Route selection in the Internet is a consequence of hand-written policies. These policies are optimized for and reflect local interests of an autonomous system (AS). However, implementation can be erroneous, leading to sub-optimal end-to-end connections [10]. Since there are almost no end-to-end guaranties for Internet, packets may not travel the shortest path to its destination. As a result, routing in the Internet decreases performance of applications.

Applications can influence their own traffic via source routing protocols [8]. However, AS'es usually run protocols like MPLS [18] which work only in a single domain. Recently, several initiatives where created to create programmable

D. an Mey et al. (Eds.): Euro-Par 2013 Workshops, LNCS 8374, pp. 333–342, 2014.

networks [12,19]. Network routers that support OpenFlow [11] technologies can allow computer programs to enforce their own forwarding rules. However, only a few network operators employ such routers, and so the control over the routing decisions remains with the operators. Furthermore, the Internet topology and its qualities at a given moment in time may only be partly known [13], making it difficult to find optimal paths. As a consequence, Internet path control is only possible for a part of a network path.

Here, we turn to cloud technology to gain control packet flows. End-users can allocate virtual resources that can be used as virtual routers. By measuring the quality of the between all virtual routers and end-points, we find the optimal path between two end points. If an path with better characteristics is found over a virtual router, we setup a tunnels and routing to control the traffic.

The automated management of such overlay networks and their respective cloud resources requires dedicated control structures. Once these control structures are designed and implemented, we need to assess to what extent cloud resources deliver significantly better results with respect to the controllability of network properties.

In this paper we introduce MeTRO, a framework of **M**anag**e**ment **T**ools for **R**outers **O**n-demand routers. MeTRO measures and setup routing in a continuous adaptive manner to adjust to change in the Internet. We evaluate our framework in a diverse setup and we find that it can improve on path latency by as much as 95%. MeTRO exhibits such latency improvement behaviour in 58% of the experiments. This reduction means that in a substantial cases that large scale distributed virtual environments [7,16] such as; multiplayer online games, distributed military simulation, and collaborative design, become more responsive.

1.1 Related Work

Savage et al. [14] have shown that even in situations with partial control and knowledge of the Internet topology, alternative paths can be created that have better qualities than those constructed with the current Internet prevailing algorithms. With a technology called Detour, Savage measured the performance of alternative paths that run via intermediate nodes that were coupled to the Internet at various locations. Ly et. al. [9] implemented a similar technology using *traceroute* traces that confirmed the observations of Savage et al. The drawback of both Detour and Ly's approaches is that it requires access to the physical machines representing the intermediate nodes. Our contribution is that we use multiple clouds and an automated framework to find and create better paths in the Internet.

This paper is organized as follows. Section 2 describes our framework for finding optimal paths and identifies three use-cases; we briefly report on related work. Section 3 presents the extensive evaluation setup and discusses the results obtained through a large number of measurements.

2 The MeTRO Framework

Here, we describe the architecture of MeTRO, our control and measurement framework, and its generic approach to cloud-hosted routers. There are currently two strategies to create optimized path in the Internet: 1) let ISPs optimize or 2) use methods such as Detour [14]. Since ISPs have an incentive to optimize themselves based on cost, the only viable option is using methods like Detour. Detour adds a fixed intermediate hop between two hosts, to gain (partial) control over the path that a packet takes in the Internet. However, it requires access to physical machines locate in different administrative domains.

Building on previous work [9,14], we present a framework that takes advantage of clouds to control as well as to optimize traffic latency. Intuitively, MeTRO is a framework that finds the optimal path between two end-points by using virtual machines (VM) residing in the cloud. The main idea it to add one hop (hosted by a VM) between two end-points and learn whether the given intermediate hop leads to a better path between the two end-points. Throughout the paper we use the term "better path" to describe a path between two points which has the lowest latency among the measured paths between these two points.

MeTRO defines two types of *agents*: 1) *agents* which perform measurements and forward the traffic, and 2) a *controller* which collects the measurement results and adjusts the paths between the agents (by setting-up tunnels and routing). An agent typically runs inside a virtual machine, therefore becoming a *virtual router*; the controller may be deployed anywhere.

Figure 1 shows the closed control-loop implemented by our MeTRO framework. The *measurement-collect-adjust* strategy allows our framework to continuously adapt to the dynamic Internet environment. During the **Setup** stage, the end-user chooses the virtual machine types and locations where to deploy each agent. Next, MeTRO will instantiate the corresponding virtual machines and collect their IPs at the controller. During the **Bootstrap** stage, the controller will copy and remotely execute the script that deploys the agent in each virtual machine. These virtual machines will now act as virtual routers. For allocating and bootstrapping the virtual machines we used [15].

Once the virtual routers are running, the MeTRO controller regularly cycles through the following three stages: **Measurement** - executes the latency measurements scripts at every agent; **Collect** - after the measurements are completed, the controller collects all data from every agent, then calculates the fastest paths between the all agents; **Adjust** - if a lower latency path is found, MeTRO will build the corresponding overlay network between the endpoints, and setup routing such that traffic is routed accordingly.

The pseudo-code in Algorithm 1 finds the optimal path between two endpoints $agent_a$ and $agent_b$ by using a RTT table and calculating all the possible paths through h agents. This RTT table is populated by the MeTRO controller via the collected measurements. The algorithm is executed by the controller. The `lat` function calculates the latency of the input path, by using the RTT table and taking into account a constant penalty p for each hop. This penalty represents

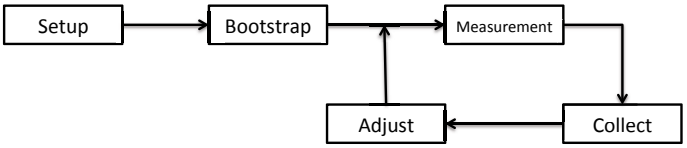


Fig. 1. The closed control-loop implemented by the MeTRO framework

the average time required to forward a packet. The `append` and `remove` functions appends and, respectively, removes the second to last element in the list.

Initially, `lst` set to $agent_a$ and $agent_b$. If multiple better paths are found, the path with the lowest latency is picked for setting up an alternative path between the end-points and dynamically create an overlay network. Currently, we only consider latency for selecting better paths, however, other criteria, such as bandwidth, jitter or a combination of such metrics, can be used to define better paths. The measurements for such network metrics could use tools like Iperf [17] or Nagios [4]. The algorithmic complexity of Algorithm 1 is $\mathcal{O}(n^h)$ where n is the number nodes in the network. While implementing Dijkstra's algorithm is more efficient, we choose this algorithm for more clarity.

Algorithm 1. findLowestPath

```
Require: iplst, lst, h, p, agent_a agent_b
  if h == 0 then
    return lst
  else
    fastest = lst
    if lat(fastest,p) > lat(direct,p) then
      return agent_a, agent_b
    end if
    for ip ∈ iplst do
      if ip ∉ lst then
        lst.append(ip)
        t = findLowestPath(iplst, lst, h-1, p, agent_a agent_b)
        if lat(t) < lat(fastest) then
          fastest = t
        end if
        lst.remove(ip)
      end if
    end for
  end if
```

2.1 Functional Scenarios

Here, we describe three different scenarios where MeTRO can be employed: 1) troubleshooting routing policies 2) reachability monitoring and 3) virtual networks with specific characteristics.

Troubleshooting Routing Policies. Our framework can also be used for finding configuration errors in routing policies. ISPs can determine if their routing policies are optimal by communicating from a given point within their own network to all the measurements nodes. Since clouds offer great flexibility, the ISP can allocate and instantiate multiple VMs around to world by employing our framework. If our measurement framework finds an alternative better path, this means that somewhere in the intermediate network a routing policy is not optimally configured. A simple *traceroute* from all nodes identifies the node where the policy is not optimal.

Clouds provide a flexible alternative to NLNOG Ring, as hosts can be created on-demand for troubleshooting and debugging purposes. In addition, the cost of on-demand virtual machines for collecting views are marginal as opposed to continuously running and maintaining servers. Collecting views using *traceroute* gives insight in how traffic is routed. If a configuration error is made, this will be detectable by comparing multiple views next to each other.

Reachability Monitoring. Our proposed measurement framework can also be used to monitor the reachability of a certain destination throughout the Internet. If a destination is unreachable, it can mean that it is unavailable for every one in the Internet. It can also mean that one of the intermediate ISP "blackholes" the traffic or hijacks the prefix. MeTRO can monitor multiple destinations and check whether specific prefixes are reachable either form inside the network, or outside the network.

Virtual Networks. Another way to use our framework is to create an overlay network using clouds. While overlay networks [6,14] and virtual networks [5] are well studied, our framework can build a virtual network dedicated to optimizing a specific metric, such as latency, bandwidth etc., over multiple ISPs.

Typical beneficiaries of such overlay networks are large scale distributed virtual environments [7,16], like multi-player online games, distributed military simulation, and collaborative design. As measurement nodes are scattered around the network, dynamic tunnels are built between these measurement nodes, and the network changes dynamically on the basis of the underlying characteristics of the ISPs.

3 Experimental Evaluation

Our goal is to assess whether cloud-hosted virtual routers are a viable solution for improving the controllability of network properties. To that end, we compare a physical network of routers to virtual routers deployed in the cloud by analyzing the quality of alternative paths found using our Algorithm 1.

3.1 Experimental Setup

Figure 2 shows an overview of our experimental setup consisting of both physical and virtual resources. The endpoints of each experiment are located in a

physical network. We have two types of experiments with respect to the intermediate (used for routing) nodes' location: (a) **physical-physical**, where the intermediate node is also located in the physical network, (b) **physical-virtual**, where the intermediate node is located in a cloud resource. For each type of experiment we also look at each Internet protocol considered: IPv4 and IPv6, where available.

The **physical-physical** experiments represent the baseline performance. Here, we performed measurements from every host to every other host (75×74 sets of results). Three round trip time (RTT) measurements where taken every five minutes, over the span of 48 hours, totaling to 129600 measurements from a given node to every other node in the physical network.

For the **physical-virtual** experiments, we selected every possible pair of hosts from the physical network and for each pair we deployed several virtual routers in each cloud zone considered.

To compute the better paths between every two endpoints, we ran our Algorithm 1 with the following parameters: the additional hop penalty $p = 0.3ms$ and the maximum number of hops $h = 8$. The algorithm uses the measurements collected from both the **physical-physical** and **physical-virtual** experiments as input. Hops considered in each experiment are in a single environment, i.e., either NLNOG Ring, BightBox or Amazon EC2.

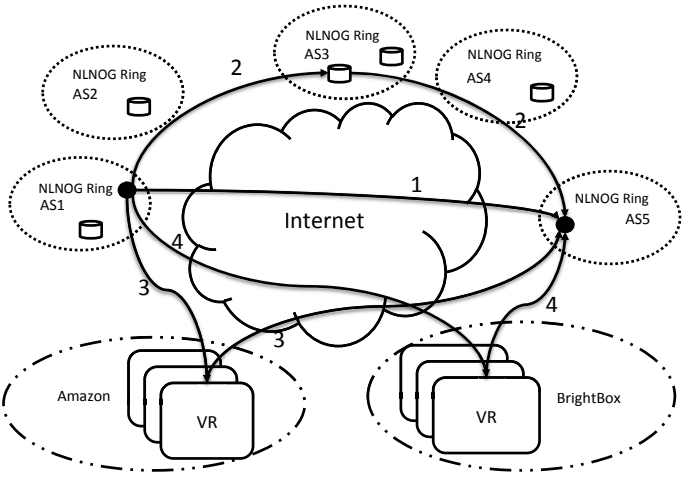


Fig. 2. Overview of our experimental setup using NLNOG Ring, Bright Box and Amazon EC2 resources. 1) a direct path between endpoints located in different ASes 2) A path with an additional hop located in NLNOG Ring 3) and 4) paths using a virtual router (`VR`).

As a physical network, we used the NLNOG Ring [3], which is an open initiative where network operators donate servers to create network debugging and monitor facilities for their networks. Each participating ISP provides a network

measurement node which allows other ISPs to perform measurement and obtain addition information about the Inetnet routing topology At the time of our experiments the NLNOG Ring consist of 75 nodes in 72 different Autonomous Systems. Servers in the ring are located in Europe(68), but also in North America(3), Asia(2), Pacific(1) and Africa(1). All the servers of the NLNOG Ring have IPv4 and IPv6 global reachable addresses.

For our **physical-virtual** experiments we used two different cloud providers: Amazon EC2 [1] and Bright Box [2]. Amazon EC2 has 10 regions, each with several availability zones. The regions are located in North America (4), South America (1), Europe (1), Asia&Pacific (4). Bright Box has one region with 2 availability zones, both being located in Manchester(GB). Amazon EC2 only provides IPv4 connectivity, while Bright Box provides both IPv4 and IPv6 connectivity.

3.2 Results and Discussion

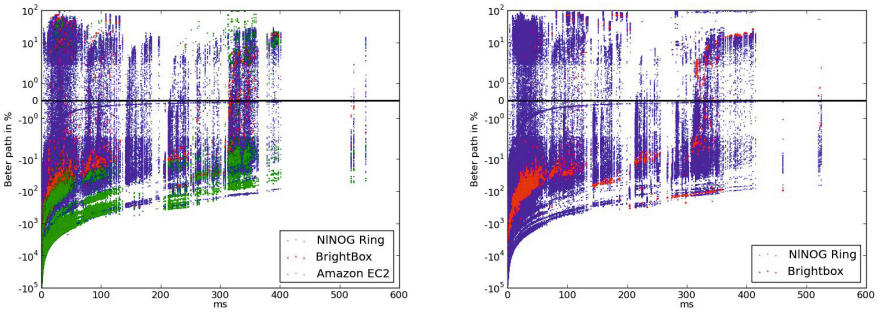
All results are collected in Table 1, Figures 5, 3 and 4. Each of them highlights different aspects of our results and we proceed to discuss each of them in more detail in the following.

Table 1. The total number of "better paths" using a different number of hops located either in the NLNOG Ring (NLNR), BrightBox(BB) or Amazon EC2(AM), for each IP(*v4) and IP(*v6)

#(Number of paths)	NLNRv4	NLNRv6	BBv4	BBv6	AMv4
Direct path Faster	2324	2405	n.a.	n.a.	n.a.
1 hop	1050	1078	515	497	1107
2 hops	1143	696	371	243	2921
3 hops	608	473			9273
4 hops	264	392			23944
5 hops	133	230			
6 hops	24	88			
7 hops	4	38			
8 hops	0	2			

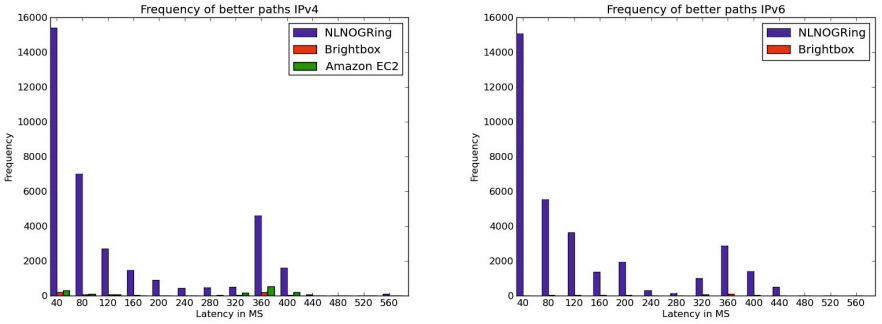
Table 1 shows the number of better paths found for each experiment type and considering $h = 0, 1, \ldots, 8$ hops. We notice that the number of better paths in the NLNOG Ring is inversely proportional to the allowed number of hops (h), while the number of better paths in Amazon is directly proportional to h. This is due to the fact that Amazon has multiple zones located geographically close to each other which results in a factorial explosion of the number of paths per region, while the latency offered by the zones in the same region is quite similar.

Figures 3a and 3b plots in log-normal scale all alternative paths found between every pair of hosts. The x-axis values represent the latency of the direct path between each pair of hosts, while each dot represents the percentage by which the alternative path is faster. Negative percentages represent slower paths.



(a) IPv4. (b) IPv6.

Fig. 3. All alternative paths using one hop and different Internet protocols



(a) IPv4. (b) IPv6.

Fig. 4. Frequency of "better paths" compared to the direct paths for different Internet protocols

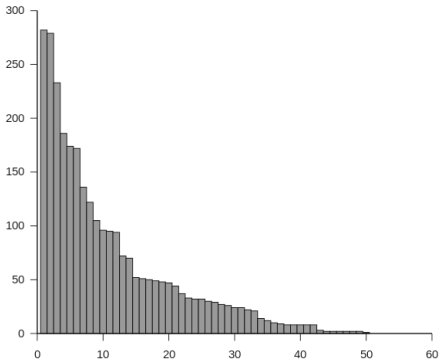


Fig. 5. Some of the NLNOG Ring nodes provide up to 10 times more "better paths" than others

Figures 4a and 4b show how many better paths are available for each latency interval (20ms). By grouping the number of alternative better paths, we can get an indication of where optimizations of paths latency can be performed.

Figure 5 shows the NLNOG Ring nodes that provide better paths. When resolving[1] the IPs of these nodes, we found that the best performing nodes are located physically close to large internet exchanges (e.g. AMS-IX, NYIX, DE-CIS, and LINX).

4 Conclusion

As the Internet optimization domain is partitioned due to the use of BGP, managing static routing policies that are applied to a substantial amount of prefixes in an dynamic environment is left to individual ISP economic priorities and prone to human error. These routing policy errors often leads to bottlenecks, increased latency or traffic can be black holed. To avoid these bottlenecks, users need control and programmability in the network.

In this paper we introduced MeTRO, a framework of tools that uses cloud infrastructure to deploy alternative, virtual routers. Within this framework, we proposed a method to explore alternative paths with specific properties, such as low latency paths. Our results are very encouraging showing that MeTRO decreases the latency in 58% of the cases studied, albeit increasing the number of hops. We showed that our framework can be used to detect unoptimized routing policies in the Internet, monitor the reachability of the Internet and create virtual networks with specific characteristics.

Acknowledgments. This work is supported by Dutch national research program COMMIT. We would like to thank Paola Grosso for reviewing this paper. In addition, we would like to thank Job Snijders and the NLNOG Ring community for providing access and resources to their NLNOG Ring servers and datasets.

References

1. Amazon EC2, http://www.amazon.com/ec2/
2. Brightbox, http://www.brightbox.com
3. The NLNOG RING, http://ring.nlnog.net/
4. Barth, W.: Nagios: System and network monitoring. No Starch Pr. (2008)
5. Chowdhury, N.M.M.K., Boutaba, R.: Network virtualization: state of the art and research challenges. IEEE Communications Magazine 47(7), 20–26 (2009)
6. Duan, Z., Zhang, Z.L., Hou, Y.T.: Service overlay networks: Slas, qos, and bandwidth provisioning. IEEE/ACM Transactions on Networking, TON (2003)
7. Funkhouser, T.A.: Ring: a client-server system for multi-user virtual environments. In: Proceedings of the 1995 Symposium on Interactive 3D Graphics. ACM (1995)

[1] IP resolved using GeoIP http://www.maxmind.com

8. Johnson, D.B., Maltz, D.A.: Dynamic source routing in ad hoc wireless networks. Kluwer International Series in Engineering and Computer Science, pp. 153–179 (1996)
9. Ly, C., Hsu, C.H., Hefeeda, M.: Improving online gaming quality using detour paths. In: Proceedings of the International Conference on Multimedia. ACM (2010)
10. Mahajan, R., Wetherall, D., Anderson, T.: Understanding bgp misconfiguration. ACM SIGCOMM Computer Communication Review 32(4), 3–16 (2002)
11. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: Openflow: enabling innovation in campus networks. ACM SIGCOMM Computer Communication Review 38(2), 69–74 (2008)
12. Meijer, R., Strijkers, R.J., Gommans, L., De Laat, C.: User programmable virtualized networks. In: Second IEEE International Conference on e-Science and Grid Computing, e-Science 2006, pp. 43–43. IEEE (2006)
13. Nakao, A., Peterson, L., Bavier, A.: Scalable routing overlay networks. ACM SIGOPS Operating Systems Review 40(1), 49–61 (2006)
14. Savage, S., Anderson, T., Aggarwal, A., Becker, D., Cardwell, N., Collins, A., Hoffman, E., Snell, J., Vahdat, A., Voelker, G., et al.: Detour: Informed internet routing and transport. IEEE Micro 19(1), 50–59 (1999)
15. Strijkers, R., Makkes, M.X., de Laat, C., Meijer, R.: Internet factories: Creating application-specific networks on-demand. Computer Networks (to appear, 2014)
16. Ta, D.N.B., Nguyen, T., Zhou, S., Tang, X., Cai, W., Ayani, R.: Interactivity-constrained server provisioning in large-scale distributed virtual environments. IEEE Transactions on Parallel and Distributed Systems 23(2), 304–312 (2012)
17. Tirumala, A., Qin, F., Dugan, J., Ferguson, J., Gibbs, K.: Iperf: The tcp/udp bandwidth measurement tool (2005)
18. Xiao, X., Hannan, A., Bailey, B., Ni, L.: Traffic engineering with mpls in the internet. IEEE Network 14(2), 28–33 (2000)
19. Yemini, Y., Da Silva, S.: Towards programmable networks. In: IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, pp. 1–11. Citeseer (1996)