Towards an Open Path Selection Architecture

Damien Saucez†, Benoit Donnet†, Olivier Bonaventure†, Dimitri Papadimitriou*
† Université catholique de Louvain – CSE Department, Belgium
* Alcatel-Lucent Bell – Antwerpen, Belgium

***Abstract*— With today's peer-to-peer applications, more and more content is available from multiple sources, increasing so the number of available paths between a source and a content. This is not however the only origin of multiple paths. Indeed, in the near future, hosts will benefit from multiple paths to reach one destination host due to the deployment of dual-stack IPv4/IPv6 hosts, but also due to new techniques such as `shim6` currently being discussed within the IRTF Routing Research Group. All these hosts will need to rank paths in order to select the best one to reach a given destination/content. In this paper, we discuss a generic service that works in any context requiring a path selection. This service is scalable, lightweight and designed to be readily deployed in ISP, corporate, or campus networks.**

I. Introduction

While, previously, a single path between two machines was assumed, we are now seeing with the evolution of the Internet topology expansion the rising of multiple paths with different performances. In such a context, it is crucial for both applications and operators to easily select the path that better suits their needs. This is clearly the issue we tackle in this paper by proposing a generic path selection service that works in any context requiring a path selection. Our service requires minimal changes (i.e., a server installation) in the current Internet architecture and implementation, is scalable, lightweight, and designed to be easily deployed in ISP, corporate, or campus networks.

In Sec. I-A, we state that the path selection issue is not only limited to peer-to-peer applications. In Sec. I-B, we review the applications and operators requirements for a path selection service. Finally, in Sec. I-C, we look over the existing solutions and explain that, despite their inherent benefits, they are insufficient for solving any path selection issue.

A. Path Selection Issues

During the last years, we have seen the emergence of technologies severely challenging three assumptions that have driven the development of most Internet protocols and mechanisms. A first assumption is that (usually) one address is associated to each host. Secondly, the forwarding of packets is often exclusively based on the destination address. For this reason, there is usually a single path between one source (or client) and one destination (or server). At last, the Internet was designed with the client-server model in mind assuming that many clients receive information from (a smaller number of) servers. During the last years, these assumptions have been severely challenged.

The client-server model does not correspond to the current operation of many applications. First, large servers are usually replicated and various content distribution networks are used to distribute content [1], [2], [3]. Second, the proliferation of peer-to-peer applications implies that most clients also act as server. This is currently creating several problems in many Internet Service Provider (ISP) networks [4]. In such a context, the client-server asymmetry does not hold anymore as earlier. For a given content, clients have to choose between multiple destination addresses.

Moreover, due to the transition from IPv4 to IPv6 many hosts will be dual-stack for the foreseeable future [5]. Furthermore, measurements show that, in today's Internet, IPv4 and IPv6 do not provide the same performance, even for a single source-destination pair [6]. This implies that, to reach a destination supporting both IPv4 and IPv6, a source can achieve better performance by selecting the stack that provides the best performance. However, today, this selection is based on simple heuristics. For instance, as highlighted by Matsumoto et al. [7], IPv6 is chosen prior to IPv4 in most of the dual-stack configurations although IPv4 offers the best performance in most of the environments [6].

In addition, an increasing number of ISPs, but also campus and corporate networks have chosen to become multihomed by being attached to two or more ISPs [8]. For these networks, multihoming offers two main benefits: technical and economical redundancy, i.e., they remain connected to the Internet even if the link that attaches them to one of their ISPs fails or if one of their ISPs becomes bankrupt. Another important benefit shown by several studies (see, for instance, Akella et al. [9]) is that multihoming allows sites to choose paths with a better quality. However, these benefits have a serious hidden cost, namely this multihoming growth is one of the main factors responsible for the growth of the BGP routing tables which could become a problem in the next future [10], [11].

During the last years, the `shim6` working group of the Internet Engineering Task Force (IETF) has been developing a new solution to the multihoming problem that relies on the utilization of multiple IPv6 addresses on each end-host. The effort within the `shim6` working group has been focused on individual hosts such as laptops attached to a CATV and an ADSL provider and not on campus and corporate networks that could even more benefit from `shim6` if their needs were adequately addressed [12].

Finally, the *Locator/Identifier Separation Protocol* [13] (LISP) is currently being discussed within the IRTF Routing Research Group (RRG) and has been proposed as one of the possible alternatives to achieve a better scaling of the Internet architecture. LISP distinguishes between identifiers and locators. The identifiers are used to identify endhosts.

The locators are assigned to ingress routers that implement the LISP tunneling scheme. LISP thus allows an identifier to be reachable through multiple locators, increasing so the number of available paths.

B. Path Selection Requirements

1) Applications: Sec. I-A told us that we are heading towards an Internet that makes available a set of paths to the host (in terms of source and/or destination addresses) for reaching, for instance, a content. In such a context, it is thus crucial for applications[1] to select the *path* (i.e., the $< source, destination >$ addresses pair) that better suits their requirements. These requirements might be expressed in terms of network performances, such as delay, bandwidth, or jitter. In addition, it would be interesting for applications to increase their reliability. Indeed, by taking into account multiple paths, whenever the current path fails, it should be possible to quickly and easily switch to another path [14]. Further investigation should reveal the real benefits or drawbacks of such a switch. Another requirement for applications refers to the cost associated to network usage. Applications might want to favor the cheapest path and, consequently, decrease their networking bills. In addition, battery-based devices would like to avoid consuming too much resources when selecting a path. This means that, ideally, the paths evaluation should be done elsewhere. Finally, applications should be able to decide by themselves which path to use, meaning that the path selection cannot be forced on them by a third party but rather suggested.

2) Operators: If it is important for applications to select a path among a set of available ones, allowing each application to make this selection by itself is not a sustainable solution (for scaling and triggered control/responsiveness and consistency/performance reasons). Instead, letting applications to cooperate with operators might lead to a win-win situation. Indeed, by helping applications to select a path, operators can meet their own requirements. The first requirement of an operator concerns the details it should reveal to allow the path selection. It is obvious that operators do not wish to disclose details on their topologies as well as their policies. Operators might also have the will to influence both incoming and outgoing traffic so that they could control their own network usage in terms of performance but also in terms of cost and traffic engineering (i.e., some links could be defined as primary links while others are backup links).

As a global requirement (i.e., it is common to applications and operators), any path selection service deployed should be scalable, should not imply a change in the Internet hardware (i.e., routers), and should not add any state information in routers. Further, it should avoid ISPs need to cooperate with each other for allowing path selection. Finally, it should be able to address nowadays path selection challenges but also future scenarios, i.e., the path selection service must be generic.

It is worth to notice that any path selection service should be able to answer the three fundamentals requirements enacted by Akella et al. [15]. Firstly, the path selection service must be able to monitor a path performance. Second, the service must be able to direct traffic over the selected path. Finally, the service must be capable of predicting the future paths behavior in order to divert traffic to the most appropriated paths.

[1]By "applications", we understand any network-based application described earlier in this section and, thus, not only peer-to-peer.

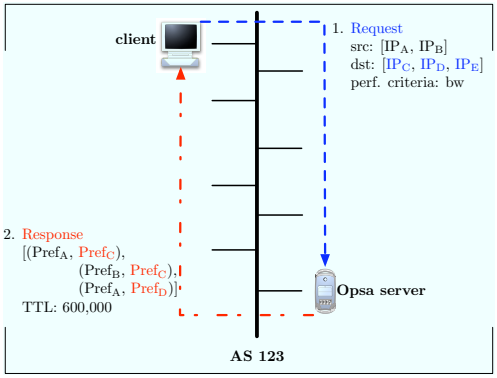

Fig. 1. OPSA request and response formats (information elements)

C. Related Work

A number of vendors tackled this path selection issue and proposed proprietary solutions, most of these being based on BGP, NAT, or rewriting [16], [17], [18], [19]. The research community also proposed solutions to similar problems. For instance, Bagnulo et al. proposed a mechanism where the source prefix of `shim6` [20] data packets is rewritten by the site routers [21]. Aggarwal et al. proposed an oracle service that would be configured by the network operator and queried by peer-to-peer applications [22]. Xie et al. propose *P4P*, a solution to give topology hints to peer-to-peer clients [23]. In P4P, ISPs maintain trackers which are contacted by peer-to-peer client to retrieve information about the topology. All these solutions mostly suffer from a lack of genericity as they all intend to solve a given path selection problem and cannot be directly applied to another problem.

II. OPEN PATH SELECTION ARCHITECTURE

In this section, we alleviate the aforementioned solutions limitations by discussing what is, to the best of our knowledge, the first generic information path selection service. We call such a service *Open Path Selection Architecture* (OPSA). This architecture is generic as it can be applied in any of the contexts depicted in Sec. I-A and aims at simplifying the path selection process. Our point with OPSA is to permit the path selection while allowing both operators and applications to meet their requirements.

OPSA works as request/response service in which a *client* sends a request to the service. The service is then in charge of selecting the server that can process the client request and reply with a response. OPSA can thus be deployed in anycast [24]. A client refers to an entity having to select a path or to rank paths, such as peer-to-peer applications, content distribution networks, but also traffic engineering capable routers (e.g., LISP [13]), or network proxies (e.g., P-Shim6 [25]). Besides, a client can have one or more addresses

such as multihomed nodes or dual-stack hosts. A server refers to the entity that is supposed to perform path ranking based on available information. The client/server model is not the only way to implement a path selection service. We however believe that it is generic enough so that it can be applied directly to several contexts (e.g., ISP, corporate, campus).

Requests sent by clients contain the following information: a list of source addresses, a list of destination addresses and an optional performance criterion (e.g., maximize the bandwidth). The OPSA server processes the request and builds a list of all possible paths based on those two lists. This possible paths list is then ranked so that the first item in the list is the most promising choice for the client while the last one is the worst. Instead of replying with complete addresses, the OPSA server can work with prefixes. Working with prefixes offers several advantages compared to full IP addresses. It first permits to reduce the replies size (e.g., a single prefix can include several addresses indicated in the request) as well as the amount of potential paths to process. Secondly, it avoids to reveal topology details and network policies to clients. Finally, when a cache is implemented, it allows a client to use a request results for several path selection problems with respect to the fact that addresses belong to the already processed prefix. Nevertheless, using prefixes has some drawbacks. First, when using prefixes as a way to aggregate information, precision can be lost (e.g., the reachability of a prefix is not the same as the reachability of an host). In addition, clients must work with end-to-end addresses which means that they must perform a best match on the returned prefixes to select the best end-to-end addresses.

In addition to a prefixes pair ranked list, the OPSA replies contain a time-to-live (TTL) information indicating how long the path ranking remains valid. This TTL is defined by network operators and is strongly dependent to the performance criterion provided by the client. When the TTL expires, it is up to the client to possibly contact the OPSA service to obtain a new path.

Considering ranked prefixes pairs list allows OPSA to hide to the client the topology and network policies. It also reduces the risk of attacks or the disclosure of sensitive information to competitors. It further allows the operators to modify the ranking algorithms according to their needs or the state of the art without disturbing clients. It thus separates the clients and operators while enabling cooperation.

The content of both request and response is illustrated in Fig. 1.

A. An implementation of OPSA

Fig. 2 shows the high level design of OPSA. It is based on three modules or engines. The first engine, named *Path Information Collector* (PIC,) is in charge of gathering various information on paths (e.g., performances, policies, costs, ...) and is described in Sec. II-A.1. Next, collected data must be efficiently represented and stored so that the processing cost of using it to rank paths is minimal. The engine in charge of representing and storing data is the *Knowledge Base* (KB – see Sec. II-A.2). Finally, OPSA must be able to combine various

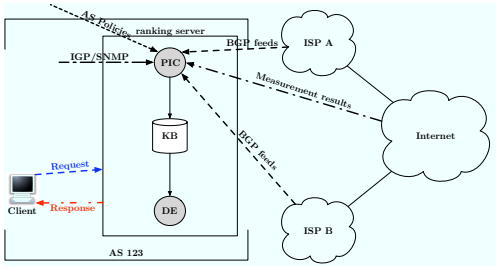

Fig. 2. OPSA general behavior

information on paths so that it can rank them. This last engine is the *Decision Engine* (DE – see Sec. II-A.3).

1) Path Information Collector: As depicted in Fig. 2, PIC collects paths information. Information are of two types: (i) administrative information, (ii) measurements information. Administrative information characterizes paths with network policies (e.g., firewalls, billing, routers graceful shutdown...) but also routing information (e.g. BGP, IGP, static routes). On the contrary, measurement information characterizes paths with their intrinsic properties. These path properties could be collected with active measurements (e.g., latency, jitter, bandwidth, path diversity) or passive measurements (e.g., TCP trace analysis, traffic matrices, SNMP).

Administrative information characterizes paths based on high-level criteria. For instance, universities are often directly connected to the national research network and use a commercial ISP for accessing the Internet. Furthermore, research networks often replicate important FTP servers like GNU/Linux distributions. Thus, when upgrading GNU/Linux hosts, universities should prefer data from the research network to commercial FTPs servers. In that case, administrative information should be sufficient to take efficient decisions. However, for paths performance dependent applications like VoIP, administrative information are not sufficient and performances information must be taken into account. Thus, in this case, the best path will be the one meeting the administrative requirements and the path performances criteria.

In addition to metrics collection, the PIC translates the different metrics into path *attributes*. Attributes are a standardized representation of the metrics, independent of their nature. The simplest way to transform metrics in attributes is to convert them into integer values. This idea comes from the LOCALPREF attribute used by BGP [26] where complex metrics are summarized as an integer. Attributes are *transitive* so that the comparison between different unrelated paths is made possible. That is, if $A > B$ and $B > C$ then $A > C$ for a given attribute. OPSA attributes have no direct meaning: a high attribute value for a path does not mean that this path is preferable to another with a lower value.

2) Knowledge Base: Once paths have been characterized, their attributes are stored in the KB. The KB might be seen as a database gathering all attributes of various paths. The KB must face two main challenges. First, it must be possible to get back any path attribute as quickly as possible. Second, given the potentially large number of paths and attributes in the KB, the KB must be as compact as possible.

3) Decision Engine: On one hand, the PIC and the KB model paths performances and properties. On the other hand, the DE compares the paths in order to select the best according to some criteria. To do so, the DE defines *Cost Functions* (CF). A CF returns the cost of a $< source, destination >$ pair (i.e., a path) for a given criterion. The cost is a numerical value characterizing a path according to one or more metrics. The cost must respect two constraints. First, the lower the cost, the better the path. Second, costs have to respect transitivity. As for attributes, transitivity is the key point of CFs as it allows one to estimate the cost of any path independently and then order them after wards. Transitivity allows caching and parallel computation of costs.

Another important point of CFs is enabling combinations to create more complicated CFs.

To ranks paths, the DE calls the appropriate CF for each possible path to rank. It then creates the ranked paths list such that the best paths are those with the lowest cost and the worst with the highest. Paths in the returned list are grouped and the list is ordered by rank. The first group of paths in the list contains all the paths with the same lowest cost value. The second group contains those with the second lowest cost and so on. For instance, a possible ordering of the ranked paths $A:1, B:2, C:1$ and $D:3$ is $(A:1, C:1, B:2, D:3)$. At that point it is important to recall that OPSA must not expose the topology and policies. Thus, the ranking cannot represent the absolute cost but the relative order of the paths with respect to to their cost. For instance, if paths A, B, C and D have a cost of $1, 4, 1$ and 7 respectively, the ranking value should be $A:1, B:2, C:1$ and $D:3$ which does not reveal the cost of the paths.

B. Path Selection Example

In the following, we present two examples of path selection algorithms. First, we present how to select the path with the more promising bandwidth. After, an example of billing based path selection algorithm is illustrated.

In the two examples, we assume that the function `update_prefix(src,dst,a,v)` tags path from *src* to *dst* with value *v* for attribute *a*. In addition, function `path_attributes(src, dst)` returns all the attributes of the path from *src* to *dst*.

The most promising bandwidth path selection algorithm is two folds. First, the PIC part is responsible of estimating the available bandwidth of the paths and to update the bandwidth information stored. If available bandwidth from *src* to *dst* is 123Mbps, the wrapper first converts the available bandwidth into a simple numerical value. Let say that by convention the bandwidth is expressed in Kbps, the value becomes 123000. If the attribute representing the available bandwidth is called **ABW**, the KB update is:

```
update_prefix(src,dst,'ABW',123000)
```

Second, the `available_bw_cf` CF is added and is such that paths with higher available bandwidth are preferred to those with less available bandwidth. The CF would be:

```
available_bw_cf(src, dst):int
begin
  attributes ← path_attributes(src, dst)
  return (MAX_BW - attributes{'ABW'})
end
```

This function shows the typical architecture of a CF. First, path attributes are retrieved. Then cost is computed as a function of the attributes. In this example, `MAX_BW` is the capacity of the best possible path in the network (typically the capacity of the best link). The cost is build as the difference between the `MAX_BW` constant and the path available bandwidth because the cost must be the smallest for the path with the higher available bandwidth. By construction, the `available_bw_cf` CF is transitive as the available bandwidth attribute is transitive.

The previous example focused on measurement information. The next example illustrates a path selection based on an administrative information. It is common for providers to charge their clients on the 95th percentile. In this example, we suppose that the client is multihomed, receives one IPv6 prefix per ISP and uses source routing. We also assume that the client already monitors the cost of its links. Then, if the cost of using ISP_A which provides prefix $2001:DB8::/48$ is $ 1,500.00, the KB is updated by:

```
update_prefix(2001:DB8::/48,::/0,'COST',2)
```

if we suppose that the attribute representing the 95th charging cost is named `COST` and only remember the ceiling value of the cost in kilo dollars. We observe that the use of prefixes instead of addresses offers a simple way to represent a large variety of paths (all the possible paths from the ISP in this example).

Lets call `minimize_cost_cf` the CF for that metric. It is implemented by:

```
minimize_cost_cf(src, dst):int
begin
  attributes ← path_attributes(src, dst)
  return attributes{'COST'}
end
```

This CF is transitive and respects the minimize costs statement.

III. CONCLUSION

In this position paper, we explained that we are going towards an Internet that makes multiple paths available between a source and a destination. We argued that the best path selection is an important issue that is not limited to peer-to-peer applications. Choosing the best path could be of interest for current applications but also under development applications.

We further explained that not only applications but also operators have requirements when selecting path. We argued that making both applications and operators cooperating for the path selection process might lead to a win-win situation, as both will meet their requirements.

We next discussed an open path selection architecture (OPSA). Respecting the OPSA architecture has different advantages. First, by definition, it agrees with the three fundamental requirements stated by Akella et al. for path selection algorithm [15]. Furthermore, it allows operators to modify path selection algorithms without having to notify clients.

Finally, OPSA is generic (i.e., it can be applied to any current and future path selection issues) scalable and supports new functionalities.

As OPSA is intended to be deployed in ISP, corporate and campus networks, it is obvious that such a service requires a certain level of standardization. Clearly, the communication protocol between the client and the service must be rigorously defined. Further, the way the PIC performs active measurements might follow principles and metrics defined within the IPPM working group.

ACKNOWLEDGEMENTS

Mr. Saucez is supported by the European-founded 027609 Agave project. Mr. Donnet is supported by the European-founded 034819 OneLab project.

REFERENCES

[1] M. J. Freedman, E. Freudenthal, and D. Maziéres, "Democratizing content pulication with coral," in *Proc. USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, May 2003.
[2] Limelight Networks, "High performances content delivery network for digital media," see http://www.limelightnetworks.com/.
[3] Akamai, "Web application acceleration and performance management, streaming media services, and content delivery," see http://www.akamai.com.
[4] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should Internet service providers fear peer-assisted content distribution," in *Proc. Internet Measurement Conference (IMC)*, October 2005.
[5] K. Cho, M. Luckie, and B. Huffaker, "Identifying IPv6 network problems in the dual-stack world," in *Proc. ACM SIGCOMM Workshop on Network Troubleshooting*, September 2003.
[6] X. Zhou, M. Jacobsson, H. Uijterwaal, and P. Van Mieghem, "IPv6 delay and loss performance evolution," *International Journal of Communication Systems*, 2007, dOI: 10.1002/dac.916.
[7] A. Matsumoto, T. Fujisaki, R. Hiromi, and K. Kanayama, "Problem Statement of Default Address Selection in Multi-prefix Environment: Operational Issues of RFC3484 Default Rules," Internet Engineering Task Force, Internet Draft (Work in Progress) draft-ietf-v6ops-addr-select-ps-05, April 2008.
[8] S. Agarwal, C.-N. Chuah, and R. H. Katz, "OPCA: Robust interdomain policy routing and traffic control," in *Proc. OPENARCH*, 2003.
[9] A. Akella, S. A., and R. Sitaraman, "A measurement-based analysis of multihoming," in *Proc. ACM SIGCOMM*, August 2003.
[10] D. Meyer, "Report from the IAB workshop on routing and addressing," Internet Engineering Task Force, Internet Draft (Work in Progress) draft-iab-raws-report-02, April 2007.
[11] T. Bu, L. Gao, and D. Towsley, "On characterizing BGP routing table growth," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM)*, November 2002.
[12] J. Schiller, "Shim6: Network operator concerns," October 2005, NANOG 35, see http://www.nanog.org/mtg-0510/schiller.html.
[13] D. Farinacci, "Locator/ID separation protocol (LISP)," Internet Engineering Task Force, Internet Draft (Work in Progress) draft-farinacci-lisp-05, November 2007.
[14] J. Arkko and I. van Beijnum, "Failure detection and locator pair exploration protocol for IPv6 multihoming," Internet Engineering Task Force, Internet Draft (Work in Progress) draft-ietf-shim6-failure-detection-11, February 2008.
[15] A. Akella, B. Maggs, S. Seshan, A. Shaikh, and R. Sitaraman, "On the performance benefits of multihoming route control," *IEEE Transactions on Networking*, vol. 16, no. 1, pp. 96–104, February 2008.
[16] Internap, "Premise-base route optimisation," 2005.
[17] Avaya, "Adaptative networking software (ANS)," 2005.
[18] Radware, "Peer director," 2002.
[19] Cisco Systems, "Optimized edge routing (EOR)."
[20] E. Nordmark and M. Bagnulo, "Shim6: Level 3 multihoming shim protocol for IPv6," Internet Engineering Task Force, Internet Draft (Work in Progress) draft-ietf-shim6-proto-09, October 2007.
[21] M. Bagnulo, A. Garcia-Martinez, and A. Azcorra, "BGP-like TE capabilities for SHIM6," in *Proc. 32nd EUROMICRO Conference on Software Engineering and Advanced Applications*, August 2006.
[22] V. Aggarwal, A. Feldmann, and C. Scheideler, "Can ISPs and P2P users cooperate for improved performance," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 29–40, 2007.
[23] H. Xie, A. Krishnamurthy, R. Yang, and A. Silberschatz, "P4P: Proactive provider participation for P2P," Yale Computer Science, YALE/DCS TR1377, March 2007.
[24] C. Partridge, T. Mendez, and W. Milliken, "Host anycasting service," Internet Engineering Task Force, RFC 1546, November 1993.
[25] M. Bagnulo, "Proxy shim6 (P-Shim6)," Internet Engineering Task Force, Internet Draft (Work in Progress) draft-bagnulo-pshim6-02, February 2008.
[26] Y. Rekhter, T. J. Watson, and T. Li, "A border gateway protocol 4 (BGP-4)," Internet Engineering Task Force, RFC 1771, March 1995.