Multi-Homed on a Single Link: Using Multiple IPv6 Access Networks

Philipp S. Tiesel
TU Berlin, Germany
philipp@inet.tu-berlin.de

Bernd May
TU Berlin, Germany
bernd@inet.tu-berlin.de

Anja Feldmann
TU Berlin, Germany
anja@inet.tu-berlin.de

ABSTRACT

Small companies and branch offices often have bandwidth demands and redundancy needs that go beyond the commercially available Internet access products in their price range. One way to overcome this problem is to bundle existing Internet access products. In effect, they become multi-homed often without running BGP or even getting an AS number.

Currently, these users rely on proprietary L4 load balancing routers, proprietary multi-channel VPN routers, or sometimes LISP, to bundle their "cheaper" Internet access network links, e.g., via (v)DSL, DOCSIS, HSDPA, or LTE. While most products claim transport-layer transparency they add complexity via middleboxes, map each TCP connection to a single interface, and have limited application support. Thus, in this paper we propose an alternative: Auto-configuration of multiple IPv6 prefixes on a single L2 link. We discuss how this enables applications to take advantage of combining multiple access networks at with minimal system changes.

1 Introduction

Multi-homed hosts or networks can be used to satisfy bandwidth and redundancy demands or to opportunistically improve network performance [1]. Indeed, today's transport layer support, e.g., via Multi-Path TCP [2, 3, 4], and SCPT [5], is moving multi-homing from corner cases [6, 7, 8] into the mainstream, e.g., in the context of IPv6 [9] and its ongoing deployment.

However, almost all cases that involve multiple hosts are today realized using proprietary middle-box solutions. These often bundle multiple cheap Internet uplinks (access networks), e.g., (v)DSL, DOCSIS, HSDPA, and LTE, and rely on L4 load balancing routers or multi-channel VPN routers. Use-cases include small company offices, branch offices, and home networks. While all solutions claim transport-layer transparency, they, typically, map each TCP connection to a single access network, do not enable applications to influence which access network to use, and increase network complexity by adding another middle-box.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ANRW '16, July 16, 2016, Berlin, Germany

© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ISBN 978-1-4503-4443-2/16/07…$15.00

DOI: http://dx.doi.org/10.1145/2959424.2959434

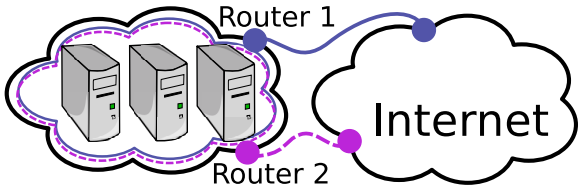

Figure 1: Multiple L3 Access Networks on a Single L2 Link.

In this paper, we propose an end-to-end alternative. Figure 1 shows an example where a home network is using two access networks each with its own router (Router 1/2). In our alternative each router announces at least one IPv6 prefix for its access network. This enables all applications within the home network the choice of access network by choosing the appropriate IP address as source. This "just" requires appropriate routing setup.

This setup enables the use of multi-path transports across both access networks, e.g., MPTCP, or multi-access enabled applications, e.g., via *Socket Intents* [10]. Socket Intents augment the BSD Socket API to enable the application to express its performance preferences and its communication pattern knowledge to bias the selection of access networks.

We realized our proposed end-to-end alternative, see Figure 1, within a Linux system. To support ease of setup, maintainability, reliability, flexibility, etc. we explored how to do this using auto-configuration rather than relying on static configuration. This allows us to determine to which extend the current RFC and their implementations enable such a scenario. We find that:

- *IPv6 Stateless Address Auto-Configuration (SLAAC)* with *Router Advertisements*, see RFC4861 [11], has support for configuring multiple prefixes and default routes as needed.
- Static address configuration with *DHCPv6* requires a single DHCPv6 server that is aware of all IPv6 prefixes from all access networks. This implies that we, currently, cannot use this setup for opportunistic bandwidth sharing, e.g. [1].
- Out of the box the support for multiple default routes and multiple DNS server configurations within the client side standard Linux utilities, e.g., `isc-dhcp-client`, `dhcpcd`, and, `network-manager`, is limited.
- Special care is needed not to interfere with DNS-based source network specific performance optimizations, such as those used by CDNs.

In the remaining sections, we outline what is needed to auto-configure multiple default routes as well as to compile appropriate DNS configurations. We then sketch what changes to the SLAAC/DHCPv6 clients are needed and how to handle multiple default routes in a way that *Socket Intents* and/or *MPTCP* can take advantage of the multiple access networks. Finally, we discuss how the client hosts gain information about the access networks and conclude.

2 SLAAC and DHCPv6 with Multiple Routers

The SLAAC protocol already supports multiple routers to send Router Advertisements (RAs) to a client to announce different prefixes and routes. It is able to resolve conflicts in most of the announced options, e.g., if the MTU of a link or the preferred lifetime of a prefix differ, the most recently announced value is considered correct [11]. However, this does not work well for some options, e.g., DNSSL and RDNSS (see Section 3). Here, a different approach is needed. Nevertheless, SLAAC supports most of the requirements to enable multi-homing on a single link.

This is not the case for DHCPv6, as specified in RFC 3315 [12]. DHCPv6 does not support configuring multiple prefixes on the same link in a decentralized fashion. After all, RFC 3315 [12] presumes the use of a single DHCPv6 server by each client at any point in time. We can circumvent this problem if we use a single—centrally maintained—DHCPv6 server that knows all prefixes for all access networks. It can then assign addresses for multiple prefixes by answering with multiple addresses and/or multiple Interface Associations. Thus, DHCPv6 works well for all scenarios where a all access network routers are jointly managed. But, at this point it does not work fully decentralized which prohibits support for use-cases such as opportunistic bandwidth sharing [1].

3 Supporting Source based DNS Replies

Nowadays, authoritative DNS servers often return different responses based on the perceived origin of the request. This is often used by CDNs for load-balancing or/and mapping users to appropriate servers and is based on elaborate network measurements and a load-based optimization. Thus, it is not sufficient to just issue a single DNS query using a single access network, rendering the DNS configuration strategy outlined in RFC 6106 [13] insufficient for our case. Rather, we need to issue a DNS query for each access network. Thereby care is needed to not mix the answers since not all CDN servers are globally reachable. We propose to *maintain a separate resolver configurations per access network prefix*. Alternatively, one can *take advantage of the DNS Client Subnet EDNS0 extension* [14] and issue a separate query for each access network while maintaining a single resolver configuration.

Both methods mentioned above can be auto-configured with SLAAC using DNSSL[1] and RDNSS[2] options as long as each router only advertises a single prefix.

However, since DHCPv6 currently does not support separate resolver configurations the DNS Client Subnet EDNS0 extension is the only option for DHCPv6.

[1]DNSSL is the list of DNS suffix domain names to use.

[2]RDNSS is the Recursive DNS Server option which contains one or more IPv6 addresses of recursive DNS servers.

4 Network Auto-Configuring for Linux

Common client-side network configuration software, such as `DHCPCD` and `network-manager`, already support the use and configuration of multiple prefixes and default routes. However, they lack the functionality to fully enable multi-homing. They do not offer applications a choice of the gateway. Rather they just add all default routes to a single routing table in the order that they receive them. Thus, in practice a single route, namely the oldest one, is used. Moreover, the DNSSL and RDNSS options are not configurable on a per prefix basis. For example, `DHCPCD` merges them per-interface.

To enable application choices we have to tackle the routing problem of choosing an access network. We see two possible solutions: *Virtual Router Functions* or *source based policy routing*. *Virtual Router Functions* allow a per-process choice of the access network for multi-access un-aware applications. *Source based policy routing* enables multi-access un-aware applications the use of MPTCP. Moreover, multi-access aware application can choose an access network on a per socket basis by binding to the appropriate source address.

Therefore, we modified `DHCPCD`: (a) to generate separate `resolv.conf` files per prefix with all information gathered via the DNSSL and RDNSS options and (b) to enable source based policy routing on a per prefix basis.

5 Access Network Characteristics

So far, we have discussed how to enable applications to use multiple access networks. This is sufficient when the characteristics of the access networks are similar. In this case, the client can use them, e.g., via MPTCP or in a round-robin fashion, and no further extensions are needed. However, this is not sufficient when the access networks have different costs and/or different performance characteristics, e.g., satellite link fallbacks, or LTE and DSL. Here, we should enable the application to make informed decisions.

To our knowledge, there is no protocol for communicating such information. Thus, we propose to communicate network characteristics by extending router advertisements with additional options. Such network characteristics can include lower bounds on the expected latency (e.g., first-hop RTT) or the smoothed average of the available up- and downstream bandwidth (e.g., derived from interface counters or lower layer protocols).

6 Summary

Using multiple IPv6 prefixes (one per uplink/access network) on a single L2 link can enable users to take advantage of multiple access networks. This can help to increase bandwidth and satisfy redundancy demands or to opportunistically improve network performance.

Minor changes to SLAAC and DHCPv6 clients are sufficient to, in principle, auto-configure our scenario with full support for MPTCP, SCTP, and multi-access aware applications. However, care is needed with DNS to avoid performance degradations. This problem can be addressed by extending the DNS resolver. If the performance/cost differences of the bundled uplinks are too large, the client should be made aware so that it react appropriately.

7 Acknowledgements

This work was supported by the DFG (Deutschen Forschungsgemeinschaft) via the Gottfried Wilhelm Leibniz-Program (FE 570/4-1).

8 References

[1] N. Vallina-Rodriguez, V. Erramilli, Y. Grunenberger, L. Gyarmati, N. Laoutaris, R. Stanojevic, and K. Papagiannaki, "When david helps goliath: the case for 3g onloading," in *SIGCOMM HotNets*, pp. 85–90, ACM, 2012.

[2] C. Paasch and O. Bonaventure, "Multipath TCP," *Queue*, vol. 12, no. 2, pp. 40:40–40:51, 2014.

[3] A. Ford, C. Raiciu, M. Handley, S. Barre, and J. Iyengar, "Architectural Guidelines for Multipath TCP Development." RFC 6182 (Informational), Mar 2011.

[4] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, "TCP Extensions for Multipath Operation with Multiple Addresses." RFC 6824 (Experimental), Jan 2013.

[5] R. Stewart, "Stream Control Transmission Protocol." RFC 4960 (Proposed Standard), Sep 2007. Updated by RFCs 6096, 6335.

[6] C. A. Sunshine, "Host Software." IEN 178, 1981.

[7] R. Braden, "Requirements for Internet Hosts - Communication Layers." RFC 1122 (INTERNET STANDARD), Oct 1989. Updated by RFCs 1349, 4379, 5884, 6093, 6298, 6633, 6864.

[8] Y. Rekhter and P. Gross, "Application of the Border Gateway Protocol in the Internet." RFC 1268 (Historic), Oct 1991. Obsoleted by RFC 1655.

[9] C. de Launois and M. Bagnulo, "The paths towards ipv6 multihoming," *IEEE Comm. Surveys and Tutorials*, vol. 8, no. 2, 2006.

[10] P. S. Schmidt, T. Enghardt, R. Khalili, and A. Feldmann, "Socket intents: Leveraging application awareness for multi-access connectivity," in *ACM CoNEXT*, pp. 295–300, ACM, 2013.

[11] T. Narten, E. Nordmark, W. Simpson, and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)." RFC 4861 (Draft Standard), Sep 2007. Updated by RFC 5942.

[12] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)." RFC 3315 (Proposed Standard), Jul 2003. Updated by RFCs 4361, 5494, 6221, 6422, 6644.

[13] J. Jeong, S. Park, L. Beloeil, and S. Madanapalli, "IPv6 Router Advertisement Options for DNS Configuration." RFC 6106 (Proposed Standard), Nov 2010.

[14] W. Kumari, tale@dd.org, W. van der Gaast, and C. Contavalli, "Client Subnet in DNS Queries," Internet-Draft draft-ietf-dnsop-edns-client-subnet-08, Internet Engineering Task Force, Apr 2016. Work in Progress.