Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 9, 2013

F. Baker
Cisco Systems

November 5, 2012

Happier Eyeballs
draft-baker-happier-eyeballs-00

Abstract

Multihoming in modern networks has problems with differing quality of routes. This note generalizes the concept of happy eyeballs across a variety of multihoming scenarios.

Table of Contents

1. Introduction: define "Multihoming"?

In the Internet, we have a variety of definitions and implementations of multi-homing.

The concept first comes up in [RFC0760], which states that "A host should also be able to have several physical interfaces (multi-homing)." [RFC0799] restates this in another way, which the author likely thought of as equivalent but given history is tantalizing: "Furthermore, hosts can be multi-homed, that is, respond to more than one address." [RFC0830] goes on to discuss a given host (as shown in its name record) being multihomed via more than one network, and [RFC0917] refers to routers as a class as "Internet hosts that are multi-homed and thus can serve as gateway." These cases all refer to an individual system as being multihomed, defining the term as implying a multiplicity of addresses or interfaces that the host might use.

The IP Version 6 Addressing Architecture [RFC4291] goes on to assert that "A single interface may also have multiple IPv6 addresses of any type (unicast, anycast, and multicast) or scope", which is to say that a particular model which is possible but unusual in IPv4 (that a given interface might have more than one address) is possible and normal in IPv6 - and as a result any IPv6 interface, by [RFC0799]'s logic, may itself be multihomed.

[RFC1164] introduces the concept of a network being multihomed. This is not absolutely new; if [RFC0830]'s host is attached to and derives addresses from multiple networks, it is a short step to assert that a network containing it is multihomed. However, it sets forth four definitions that have proven seminal in operational deployment:

"Autonomous System" or "AS" a set of routers under a single technical administration, interconnected to other networks using an exterior gateway protocol (in this case BGP), that appears to other ASs to have a single coherent interior routing plan, and presents a consistent picture of what networks are reachable through it. [author's restatement of a larger paragraph]

Stub AS: an AS that has only a single connection to another AS. Naturally, a stub AS only carries local traffic.

Multihomed AS: an AS that has more than one connection to other ASs, but refuses to carry transit traffic.

Transit AS: an AS that has more than one connection to other ASs and is designed (under certain policy restrictions) to carry both transit and local traffic.

[RFC6555] goes on to describe a scenario in which a host has multiple addresses and is therefore, by [RFC0799]'s logic, multihomed - but with a wrinkle. In this case, some of the addresses might be IPv4 [RFC0791] addresses, and some might be IPv6 [RFC2460] addresses, and they might be on the same or different interfaces.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Generalizing RFC 6555

[RFC6555] makes it fairly clear that the applications it is thinking of are wide-ranging, referring to "(e.g., web browsers, email clients, instant messaging clients)". However, its examples and text mostly refer to web browsers, which in turn run on TCP. As a result, some commentators have inferred that it only deals with web browsers or only with TCP, and have gone on to repeat the specification for other applications. This was not the intention of the authors, and is unfortunate. But the specification does concern itself primarily with Dual Stack networks and hosts and applications in them. The best application is wider yet. This is to multihoming in all of its forms.

2.1. Implications of multihoming

A key implication of multihoming, regardless of the type of multihoming in view, is that there is potentially more than one route between two interfaces or systems, and the routes may have differing characteristics. If a host has multiple interfaces, they are often of differing technology, such as Ethernet and WiFi or WiFi and LTE, and those interfaces may have differences in delay, capacity, loss rate, monetary cost, or routing policy. Multihomed addressing implies multiple routes between systems, even if only in the first or final hop. In addition, routing information or policy may come into play;

- o [RFC6724] wonders whether any form of network address translation is in use,

- o ULAs [RFC4193] may be preferred within a domain but not advertised between domains,

- o a network may be using a prefix or address family internally but not externally,

- o a filter may be in place in a firewall, or

- o a normally-available route may be in flux or temporarily inoperative.

In the extreme, routing may offer multiple paths between two AS's, and the differences among them may be satellite vs. terrestrial paths, which differ markedly in end-to-end delay, or between fiber and microwave, which can differ markedly in loss rate.

In short, a {source, destination} address pair implies a route between those addresses, and multiple address pairs may imply multiple routes with different characteristics.

2.2. The implications of multihoming for applications

As such, the implication of multihoming for applications is that any application that finds itself on a multihomed host or attempting to open a session with a multihomed host - a host that has more than one address regardless of address family or underlying hardware - SHOULD be prepared to use any or all of them at any time, and SHOULD organize its algorithms to find one among them that provides adequate service with the least effort on its own part and impact on its user. If the application operates without user intervention or awareness, such as SMTP between MTAs, the decision should be transparent and immaterial to any user; one among the better routes should be chosen, and a user should not perceive non-intrinsic variability among them. If the application operates with user involvement, such as voice or video on RTP, web browser access, or electronic mail, the decision should be performed in a manner that minimizes user impact, including connection setup delays.

An implementation MAY prefer IPv6 routes over IPv4 routes, such as by trying IPv6 addresses first. If the difference is known, an application is well-advised to prefer native routes over translated routes, as is suggested in [RFC6724], for reasons such as those mentioned in [RFC2993].

When comparing operational characteristics such as route quality, it is difficult to make a single recommendation for all applications, as applications may have differing requirements. [RFC6555] suggests initiating a set of connection attempts at some cadence, and

accepting the first to succeed, which at least obtains a route that provably works and may have a lower round-trip-time than other choices.

If the implementation caches session metadata across multiple sessions, it may be useful to include and consider information such as

- o Some sense of the success rate in connecting to a given address or prefix, such as succeeding in N out of M attempts. M=0 implies that the address or prefix has not been tested.

- o The round trip time on successful connection set-up, allowing the implementation to try lower-RTT routes first.

o If the connection moved at least a stated among of data to or from the peer, the effective throughput achieved, enabling an implementation to select high throughput routes first. Short exchanges that do not achieve sustained throughput are likely to produce unreasonable numbers, and should therefore not be included in such a cache.

   In TCP terms, TCP's estimate of its throughput rate is the final value of cwnd*pmtu/mean_rtt times a scaling constant.

2.3. The implications of multihoming for operating systems

An unfortunate characteristic of the current Socket API [RFC3493][RFC5014] is that it forces the application to have knowledge of the IPv4 or IPv6 address of its peer and potentially of itself. While there are cases in which it is useful to know that address, such as for logging, it is seldom required or useful once the session has been established. A side-effect of having the application store the address as a result of one call and then use it in another is that the process of updating systems to use IPv6 forces a change to every implicated application. As a result, we find operating systems deploying technologies such as Bump-in-the-Host [RFC6535] APIs, which translate IPv4 system calls into IPv6 behavior without informing the application. Many of the issues encountered in renumbering [RFC4192] also derive from application shortcuts such as directly referring to addresses rather than names, or not updating cached information when it expires.

It would be better if the socket "connect" API accepted a character string, which might contain a DNS name, an IPv4 address literal, or an IPv6 address literal, negotiated the connection in an [RFC6555]-compliant manner, and then returned either a connected socket or an error code, and additionally provided an API that enabled a

application to obtain its peer's address from a connected socket, ideally as a character string. In this way, the application using a network can be independent of the network, and insulated from changes to the network layer.

Ideally, the API also enables the specification of the transport protocol or at least the type of service it seeks. It might enable an octet stream interface using TCP [RFC0793], or a dynamic set of message stream interfaces associated with one peer using SCTP [RFC4960][RFC5061]. On a "listener" interface, the Socket API should be able to do the counterpart, permitting This has two values; the socket interface it replaces permits specification of the underlying transport, and support for more modern transports enables their convenient use. This would be useful, for example, in obviating the issues of head-of-line blocking among exchanges in a pipelined TCP, obtaining web objects or performing Map/Reduce message exchanges in parallel instead.

Creation of such an API in the ANSI standard would allow the Socket API to remove gethostbyname(), with its IPv4-only limitation, and getaddrinfo(), whose use results in the issue [RFC6555] addresses.

One example of such an API, which unfortunately always uses TCP, is the java.net.Socket class

```
public Socket(String host, int port) throws IOException
public InetAddress getInetAddress()
public InetAddress getLocalAddress()
public int getPort()
public int getLocalPort()
```

The Windows WSAConnectByName and StreamSocket.ConnectAsync APIs, and the MacOSX CFStreamCreatePairWithSocketToHost API, are also examples of such an API, with the same limitation regarding the transport service.

3. Conclusions

In summary, although [RFC6555]'s examples are largely drawn from TCP and web service, is best understood as generalizing to all applications and transports. It comments on session initiation in what this note points out are essentially multihomed environments. Whenever there exist a multiplicity of possible routes, selectable by the session originator through its choice of {source, destination} address pair, the application is best served by finding a useful route, or set of routes, quickly. It SHOULD do so.

4. IANA Considerations

This memo asks the IANA for no new parameters.

5. Security Considerations

This note makes no observations regarding security, and does not impact it.

6. Privacy Considerations

This note makes no observations regarding privacy, and does not impact it.

7. Acknowledgements

The author received comments from Dan Wing, Dave Thaler, Erik Nordmark, Mark Townsley, Ralph Droms, and Stuart Cheshire.

8. Change Log

Initial Version: November 2012

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, April 2012.

9.2. Informative References

[RFC0760] Postel, J., "DoD standard Internet Protocol", RFC 760, January 1980.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC0799] Mills, D., "Internet name domains", RFC 799, September 1981.

[RFC0830] Su, Z., "Distributed system for Internet name service", RFC 830, October 1982.

[RFC0917] Mogul, J., "Internet subnets", RFC 917, October 1984.

[RFC1164] Honig, J., Katz, D., Mathis, M., Rekhter, Y., and J. Yu, "Application of the Border Gateway Protocol in the Internet", RFC 1164, June 1990.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2993] Hain, T., "Architectural Implications of NAT", RFC 2993, November 2000.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, February 2003.

[RFC4192] Baker, F., Lear, E., and R. Droms, "Procedures for Renumbering an IPv6 Network without a Flag Day", RFC 4192, September 2005.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.

[RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, February 2006.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC5014] Nordmark, E., Chakrabarti, S., and J. Laganier, "IPv6 Socket API for Source Address Selection", RFC 5014, September 2007.

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, September 2007.

[RFC6535] Huang, B., Deng, H., and T. Savolainen, "Dual-Stack Hosts Using "Bump-in-the-Host" (BIH)", RFC 6535, February 2012.

[RFC6724] Thaler, D., Draves, R., Matsumoto, A., and T. Chown,

"Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, September 2012.

Author's Address

Fred Baker
Cisco Systems
Santa Barbara, California 93117
USA

Email: fred@cisco.com