Multihoming and Applications

draft-nordmark-multi6dt-refer-00.txt

Erik Nordmark
erik.nordmark@sun.com

Multi6 design team: J. Arkko, I. Van Beijnum, M. Bagnulo, G. Houston, E. Nordmark, M. Wasserman, J. Ylitalo

Background

- Scalable IPv6 site multihoming
  - Site connected to more than one ISP
  - Scalable implies not dumping the problem in the routing system with one route per multihomed site
  - Implies each site receives an IPv6 address prefix from ISP
- Keep the IPv6 socket API stable; 128 bit addresses
- Design team has been looking at a L3 shim
- But the application issues are not specific to that approach

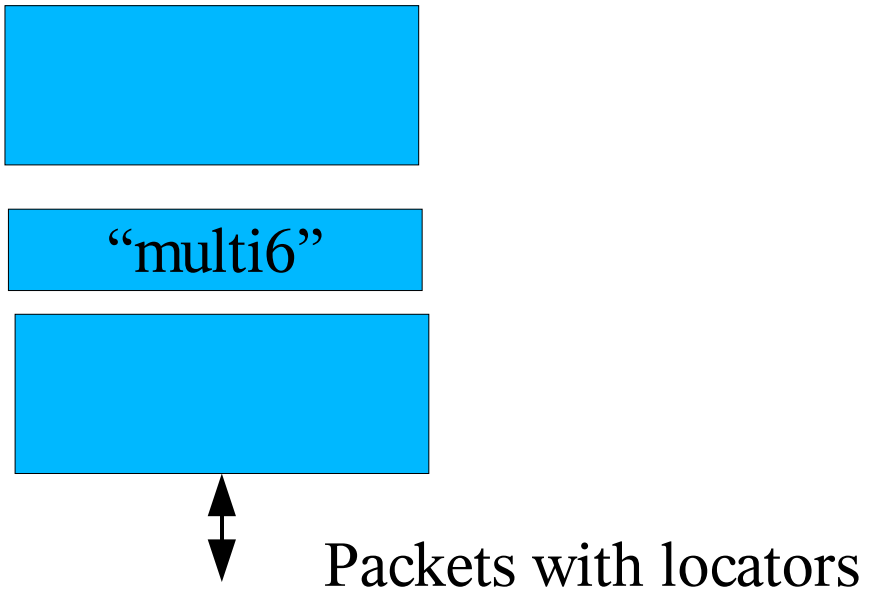
Possible solution approaches

1. Do nothing
2. Only worry about the problem during connection establishment; choose a working locator pair
3. Introduce multiple locators and a sub-layer in the stack which will make transport communication survive by being able to switch between the locators
4. Introduce a new identifier name space with a distributed system for mapping from identifiers to the current set of locators

Implications

- In #3 and #4 the applications (at the socket API) see some 128-bit quantity
  - We call this the ULID
  - Underneath there are multiple 128-bit locators
- The ULID could be one of the locators (#3), or
- The ULID could be something which isn't reachable (#4)
- Whether it is reachable or not isn't central to this discussion

IPv6 socket API



Likely outcome

- A new identifier name space would either
  - Take a long time to define, implement and deploy
  - Use the DNS AAAA and PTR records
    - Implies a hierarchical allocation i.e., a managed ID space, which will probably require some fees
    - But desire to provide multihoming benefits without registering in a managed space
- Thus multiple locators without any new ID name space is the likely outcome for at least the short and medium term

The good news

- No change for applications which use the IP address as a “short term” handle
  - Take result of getaddrinfo() and pass it to connect() or sendto()

Other application usage

- "Long-term handle"
  - Retain/cache for communication in the future
- "Callbacks"
  - A connects to B; B retains A's IP address; later B connects to A
- Referral
  - Pass address of self or of peer to a 3rd party
- "Identity Comparison"
  - Use IP address to check if peer is the same as before
  - Does anybody do this?

Possible application approaches

- Use FQDNs wherever possible
  - But not possible if e.g., no FQDN assigned to host, FQDN for service instead of host, etc.
- Use a single IP address
  - Works as long as that locator is reachable; would not benefit from the redundant paths present with multihoming
- Use the set of IP addresses aka locators
- Use the set of locators plus the ULID

Recommendations

- Move applications to use FQDNs if possible
  - Not possible in all cases due to hosts not having/knowing their FQDN, FQDNs for service and not for host, etc.
- Use set of locators (+ULID?) instead of a single locator
  - Need new API calls?
    - getlocallocators(int socket, struct sockaddr *[], int *naddr);
    - getpeerlocators(int socket, struct sockaddr *[], int *naddr);
    - setpeerlocators(int socket, struct sockaddr *[], int *naddr);

Open Issues

- Make it clear that application doesn't interpret the locator set
- Does it apply to single IPv4 + single IPv6 address case?
- Clarify security issue around identity comparison?
- Clarify RFC 3041 and DNS
- If we will tweak the API, should we add a
  - connect_to_name(, char *fqdn, int port);
  - If so, what about UDP?

Questions?