

# Stub Networks Use Case: Thread Border Router

Ted Lemon <[elemen@apple.com](mailto:elemen@apple.com)>

# Why a stub network?

- Thread uses 802.15.4 to build a mesh network
  - 802.15.4 is a constrained network technology
  - Low power (devices can run on battery for months/years)
  - IPv6-only
  - Can't be bridged to infrastructure because
    - 802.15.4 is too slow
    - We don't want all that traffic anyway because it would drain batteries
    - Layer 2 header format is incompatible

# Requirements

- Lots of devices on Thread network
- Generally these are devices that we need to control
- That means they are providing a service: their controllability
- Also sensors, which provide a service: the sensor output
- These devices need to be discoverable/operable from ordinary devices like cell phones, home speakers, set top boxes, etc.
- Don't want a special ALG between Thread and IP because
  - Experience is that you wind up with many such devices
  - Such devices prevent permissionless innovation
- Need access to cloud for firmware updates

# History

- Prior to the stub network model, there was an OpenThread border router that used proxy ND to enable communication, but did not provide discoverability
- Apple released the first stub-network-style Thread Border Router in Fall of 2020: the HomePod mini
- Support for stub router functionality showed up in OpenThread sometime after that
- More recently, quite a few vendors have shipped Thread Border Routers based on the stub router model

# Thread 1.3.0

- Thread 1.3.0 includes full stub router support
- There are differences that are probably worth thinking about as we proceed with the stub router specification
- Types of devices:
  - Home Routers
  - Set-top boxes
  - WiFi Speakers
  - Home management touchscreen devices
  - Light Bulbs
  - ???

# Managing addressability on AIL

- Thread uses XPANID to generate the AIL prefix
- This means that if there is only one Thread network connected to a particular AIL, and there is no IPv6 service on the AIL, the AIL IPv6 on-link prefix is stable
- If there is more than one Thread network, we use the “numerically lowest wins” election algorithm to pick which XPANID gets used.
- As long as we have one thread BR connecting the winning Thread network to the AIL, the AIL prefix is stable

# Managing addressability on the stub network

- This is pretty much the same.
- Thread does not have RAs
- So we update the Thread network data
- Thread network data is very constrained, so we try to keep the OSNR prefix stable

## Maintenance across stub router restarts

### Generating a ULA to provide reachability

- Because the AIL prefix is based on the XPANID, we don't need to do anything special to keep it stable across restarts—the XPANID is already being maintained that way.
- On the Thread mesh, each border router has its own ULA prefix. One border router publishes
- This means that the OSNR prefix can change over time as devices are power cycled/shut off.
- We see problems because of this. Would be nice to do something about it.
  - E.g., renumbering means that every SRP advertisement has to be renewed, wasting battery and creating mDNS traffic on infrastructure

# Using DHCPv6 Prefix Delegation to acquire a prefix to provide addressability

- We do do this, and it seems to work
- Currently only one BR will publish an OSNR prefix, and we do not try to maintain a stable prefix across BRs
- We see some uptake of this in the field, but it's not the common case
- We have seen bugs in home router implementations, e.g. a home router offering the prefix published on-link on the AIL in a DHCPv6 PD, resulting in complete routing fail

# Managing reachability (on the adjacent infrastructure link) (on the stub network)

- On the ALL we use RAs. This mostly works fine
- We do see issues when the ALL prefix isn't stable
- We have gotten some feedback asking about these weird new RAs, but mostly silence—it seems to Just Work
- Thread's network data is constrained, so we have a very simplified approach to advertising routes. We do not advertise explicit routes. We either advertise a default route, or a route to fc00::/7.
- We explicitly don't support multiple ALLs at present

# Providing discoverability between stub network links and infrastructure network links

- For discoverability of Thread hosts by AIL hosts, we provide mDNS via SRP + an advertising proxy
- This seems to work pretty well, although we've noticed that proxying a large database of services does seem to cause some multicast issues on some infrastructures
- We would like to move more toward unicast, but this isn't a solved problem yet.
- Thread hosts use discovery proxy to discover AIL hosts
- Thread hosts use SRP database (no mDNS) to discover thread hosts

# Providing reachability to IPv4 services to the stub network

- Thread BRs provide NAT64 when none is present on infrastructure
- NAT64 prefix is advertised on Thread network so that Thread hosts can do DNS64 synthesis themselves
- The DNS resolver on the Thread network does not do DNS64 synthesis
- Thread BRs monitor ipv4only.arpa to discover infrastructure prefix
- Recently we specified that PREF64 should be used when present, but I don't know if this is implemented

# Handling partitioning events on a stub network

- We used to have a really elaborate scheme for detecting and managing this
- We now use a very simple model, which is pretty much what's in the stub network document:
  - just publish an OSNR prefix when there isn't one.
  - When a partition forms that doesn't include the router currently advertising the OSNR prefix, the prefix will eventually go away
  - At this point a BR that is on the partition will wind up publishing a prefix
  - Downside to this is that there's a gap where devices on the new partition may not be reachable

# Questions?

- Bueller?