# Thread 1.3.0 Features White Paper

## July 2022

This Thread Technical white paper is provided for reference purposes only. The full technical specification is available publicly. To join gain access, please follow this link: https://www.threadgroup.org/ThreadSpec.

If there are questions or comments on these technical papers, please send them to help@threadgroup.org.



# Thread 1.3.0 Features White Paper

## July 2022

Revision History

| Revision | Date | Comments |
|---|---|---|
| 1.0 | July 19, 2022 | Public Release |

# Table of Contents:

# 1. Authors: Jonathan Hui, Sujata Neidig, Alan Collins

# 2. Introduction

Thread is a low-power wireless mesh networking protocol based on the universally-supported Internet Protocol (IP), and built using open and proven standards. A key part of a Thread mesh network is a Border Router. A Thread Border Router connects Thread networks to other IP-based networks, such as Wi-Fi or Ethernet. Like an access point does for Wi-Fi, a Thread Border Router extends connectivity to other IP-based networks and the Internet, eliminating the need for proprietary hubs or bridges. Once a Thread Border Router is connected to the network, all Thread-enabled devices can be securely accessed from smartphone, smart speaker or tablet applications, or from a cloud-based online service.

Thread was designed from the ground up to support the Internet Protocol (IP), with a goal of making it easier to integrate with existing IP-based networks. Thread 1.3.0 fully achieves this goal by standardizing Thread Border Routers and how Thread networks can connect to existing IP-based infrastructure. Thread 1.3.0 also includes a number of enhancements that improve the overall scalability, reliability, and robustness of Thread networks while maintaining backward compatibility. This whitepaper provides an explanation of the targeted marketing use cases and then a technical overview of the new features introduced in the Thread 1.3.0 Protocol Specification.

# 3. Thread 1.3.0 Use Cases

Thread 1.3.0 addresses four primary use cases:

- Enabling Matter
- Standardizing Border Routers which Connect Thread Networks to Existing IP Infrastructure
- Simplified in-field firmware updates

## 3.1. Enabling Matter

Matter and Thread are built around the same IPv6 foundation. Thread 1.3.0 enables Matter devices to join users' IP-based networks while enabling low power, long-range use cases, and brings the full functionality of IP routing and service discovery to Thread networks, enabling Matter to operate seamlessly on Thread networks.

## 3.2. Standardizing Border Routers which Connect Thread Networks to Existing IP Infrastructure

A primary goal of Thread 1.3.0 is make it simple for Thread devices to attach to existing IP infrastructure like Wi-Fi or Ethernet. Thread 1.3.0 Border Router Extensions standardizes the Border Router implementation, making it simple for Thread devices to communicate with devices outside the Thread network. Like Wi-Fi access points, Thread Border Routers provide functionality such as routing traffic and cloud connectivity, but are unique in that they can be built into existing devices from any company, thereby minimizing the need for additional dedicated hardware.

Thread 1.3.0 allows Thread devices to appear on an existing Wi-Fi network just like any other host on that Wi-Fi network. This includes allowing Wi-Fi devices to discover services on the Thread network using standard protocols like DNS-SD/mDNS. When a Wi-Fi host wants to connect with a discovered IPv6 address on Thread, it can simply send an IPv6 packet to that IPv6 destination.

## 3.3. Simplified in-field firmware updates

Thread 1.3.0 adds a new bulk transfer capability, which makes it possible to quickly and automatically update firmware on Thread devices without impacting responsiveness of controls or timeliness of event delivery. TCP is well suited to efficient bulk transfer of data and has specific advantages over UDP-based protocols, including ability to support network address translators (NATs) and firewalls. One clear advantage is that a TCP connection keeps a hole open in a network address translator (NAT) or firewall. Thread 1.3.0 standardizes the use of TCP in Thread networks, providing a simple mechanism for firmware updates.

# 4. Thread 1.3.0 Features

To support the target use cases, Thread 1.3.0 introduces the following new features:

- **IPv6 connectivity** across Thread Border Routers, including automatic configuration of IPv6 prefixes and routes across both the Thread network and adjacent IP network.
- **DNS-based Service Discovery**, including the use of DNS-SD Service Registration Protocol (SRP) to avoid multicast on the Thread network, Advertising Proxy for DNS-SD Service Registration Protocol to publish Thread network services on the adjacent IP network, and Discovery Proxy for Multicast DNS-Based Service Discovery to allow Thread devices to discover services on the adjacent IP network.
- **Transport Control Protocol (TCP) requirements**, including mandatory TCP options that Thread devices must include in their TCP implementations to ensure performant operation.

Thread 1.3.0 features are mapped into applicable Thread devices roles in the table below. The subsections provide more details on these features.

| **Feature** | **Border Router** | **End Device** |
|---|---|---|
| Bidirectional IPv6 reachability | X | |
| DNS-Based Service Discovery<br>• Service Registration Protocol<br>• Advertising Proxy<br>• Discovery Proxy | X | X |
| Bulk Transfer<br>• MAC Updates<br>• Delay-aware Queuing | X | X |
| Bulk Transfer<br>• TCP/TLS | | X |

## 4.1. IPv6 Connectivity

Thread Border Routers are responsible for providing IPv6 reachability between a Thread network and an *infrastructure network*, which is the primary network infrastructure present in a home or commercial building. A typical infrastructure network may be composed of one or more Wi-Fi and Ethernet links, but also includes any reachable IP network, including the public Internet. The *Adjacent Infrastructure Link* is the IP link that is directly connected to the Thread Border Router. Many home networks consist of a single Wi-Fi network and the Adjacent Infrastructure Link refers to that Wi-Fi network.

Support for IPv6 is widely available in modern devices, but not all networks have routable IPv6 configured. All IPv6 devices will self-configure an IPv6 link-local address regardless, and this is used for one-hop communication between nodes on the same link. However, for communication between different links, a routable IPv6 address is required. Since the Thread Network is not the same link as the Adjacent Infrastructure Link, routable IPv6 addresses are required for communication between the Thread Network and devices on the Adjacent Infrastructure Link.

### 4.1.1. IPv6 Prefix Configuration on Thread Network

Using the Thread Network Data, a Border Router configures Thread Devices with an Off-Mesh Routable (OMR) address. An OMR address may be an IPv6 Global Unicast Address (GUA), an
IPv6 Unique Local Address (ULA), or some other routable IPv6 unicast address type defined in future IETF specifications.

By default, a Thread Border Router uses DHCPv6 Prefix Delegation to request an IPv6 prefix from the Adjacent Infrastructure Link [RFC 8415]. If the Thread Border Router is not able to obtain an IPv6 prefix from the infrastructure network, the Thread Border Router will generate a /64 ULA prefix for the Thread Network.

### 4.1.2. IPv6 Prefix Configuration on Adjacent Infrastructure Link

If a router is already providing a usable IPv6 prefix on the Adjacent Infrastructure Link, the Thread Border Router refrains from providing routable unicast IPv6 addresses to IPv6 devices on the Adjacent

Infrastructure Link. Existing routers advertise prefix information in Router Advertisement (RA) messages using a Prefix Information Option (PIO). A usable prefix has both the on-link and autonomous address-configuration flags set, which enables IPv6 devices on that link to configure and use SLAAC-configured routable unicast IPv6 address for communication with devices outside that link (such as devices attached to the Thread Network served by the Border Router).

If a Border Router finds no Router Advertisements on the Adjacent Infrastructure Link containing a suitable PIO, it generates a ULA prefix using the 40 most-significant bits of the Thread Network's Extended PAN ID as the global ID. The Border Router then generates a /64 subnet prefix from that ULA prefix by setting the 16-bit subnet ID to the 16 least-significant bits of the Extended PAN ID.

Including this PIO allows other IPv6 devices attached to that Adjacent Infrastructure Link to use Stateless Address Autoconfiguration (SLAAC) to configure one or more routable unicast IPv6 addresses for themselves, which they can use for off-link communication with Thread Devices attached to the Thread Network on the other side of the Border Router.

IPv6 prefix configuration by the Border Router is dynamic, and may change over time. If, upon startup, the Border Router finds no suitable IPv6 Router Advertisements, then it emits its own PIO to configure hosts on the Adjacent Infrastructure Link as described above. If a Border Router that is emitting its own PIO later observes suitable IPv6 Router Advertisements coming from some other IPv6 router, it stops sending its own. Similarly, if a Border Router that is not emitting PIOs later observes that no suitable IPv6 Router Advertisements coming from other IPv6 routers, then it begins sending its own. This process of tracking changing network state continues for as long as the Border Router is running.

### 4.1.3. Routing to Thread Network

A Thread Border Router announces reachability of the OMR Prefix by sending IPv6 ND Router
Advertisement (RA) messages on its Adjacent Infrastructure Link. These RA messages contain an IPv6 Route Information Option (RIO) indicating that this particular IPv6 prefix is reachable via this particular Border Router [RFC 4191].

Other IPv6 devices on the Adjacent Infrastructure Link receive this IPv6 Route Information Option and record the information in their IPv6 routing table. The RIO indicates to them that IPv6 packets destined to IPv6 addresses within the OMR Prefix should be sent to the Thread Border Router for forwarding, rather than being sent to their usual IPv6 default router (if present).

### 4.1.4. Routing to Adjacent Infrastructure Link

Thread Border Routers advertise routes to other IPv6 networks in the Thread Network Data. If the Thread Border Router has a default route to the network infrastructure, it will advertise a default route in the Thread Network Data.

To support scenarios where a Thread network has multiple Thread Border Routers attached to different Adjacent Infrastructure Links, Thread Border Routers will also advertise more-specific routes to prefixes that they learn them on the Adjacent Infrastructure Link. Advertising a more-specific route to the IPv6 Prefix learned from PIOs on the Adjacent Infrastructure Link allows Thread devices to communicate with hosts on that Adjacent Infrastructure Link, as well as with different Thread networks.

When a device on the Thread Network has an IPv6 packet to send, addressed to an IPv6 destination address outside the Thread Network (e.g., a smart home accessory on the Thread Network replying to a command received from a smartphone), the IPv6 packet is forwarded through the Thread Network using the normal Thread Network routing mechanisms until it reaches the appropriate exit (egress) Thread Border Router for that IPv6 destination address.

## 4.2. DNS-Based Service Discovery

DNS-Based Service Discovery [RFC6763] is a component of Zero Configuration Networking [RFC6760] [ZC] [I-D.cheshire-dnssd-roadmap]. DNS-based Service Discovery (DNS-SD) allows services to advertise the fact that they provide service, and to provide the information required to access that service. DNS-SD clients can then discover the set of services of a particular type that are available. They can then select a service from among those that are available and obtain the information required to use it.

Hosts on typical infrastructure networks like Wi-Fi and Ethernet use Multicast DNS (mDNS) [RFC 6762] to publish their services. However, multicast and broadcast are inefficient on wireless mesh networks like Thread. Although DNS-SD using the DNS protocol (as opposed to mDNS) can be more efficient and versatile, it is not common in practice, because of the difficulties associated with updating authoritative DNS services with service information.

To facilitate fast, reliable, service discovery on Thread Networks, all Border Routers are required to implement three related capabilities in support of bidirectional unicast service discovery:

1. Thread Service Registry
2. Advertising Proxy
3. Discovery Proxy

### 4.2.1. Service Registry

The Thread Service Registry consists of the following IETF technologies:

- **DNS recursive resolver** to answer queries for all valid DNS record types, including hostname records, for example, DNS type "A" and "AAAA" address records [RFC 1034] [RFC 1035].
- **Service Registration Protocol (SRP) server** for DNS-Based Service Discovery, which accepts DNS record registration requests from clients that want to advertise their services [draft-ietf-dnssd-srp].
- **DNS authoritative server** that answers authoritatively for DNS-Based Service Discovery records [RFC 6763] and any other DNS records registered with the Thread Service Registry by clients using SRP.

SRP uses the standard DNS Update [RFC2136] mechanism to enable DNS-Based Service Discovery using only unicast packets. This makes it possible to deploy DNS Service Discovery without multicast, which greatly improves scalability and improves performance on networks where multicast service is not an optimal choice, such as on Thread networks.

Existing practice for updating DNS zones is to either manually enter new data, or else use DNS Update [RFC2136]. Unfortunately DNS Update requires either that the authoritative DNS server automatically trusts updates, or else that the DNS Update client has some kind of shared secret or public key that

is known to the DNS server and can be used to authenticate the update. Furthermore, DNS Update can be a fairly chatty process, requiring multiple round trips with different conditional predicates to complete the update process.

The SRP protocol adds a set of default heuristics for processing DNS updates that eliminates the need for DNS update conditional predicates: instead, the SRP server has a set of default predicates that are applied to the update, and the update either succeeds entirely, or fails in a way that allows the registering service to know what went wrong and construct a new update.

SRP also adds a feature called First-Come, First-Served Naming, which allows the registering service to claim a name that is not yet in use, and, using SIG(0) [RFC2931], to authenticate both the initial claim and subsequent updates. This prevents name conflicts, since a second SRP service attempting to claim the same name will not possess the SIG(0) key used by the first service to claim it, and so its claim will be rejected and the second service will have to choose a new name.

Finally, SRP adds the concept of a 'lease,' similar to leases in Dynamic Host Configuration Protocol [RFC8415]. The SRP registration itself has a lease which may be on the order of an hour; if the registering service does not renew the lease before it has elapsed, the registration is removed. The claim on the name can have a longer lease, so that another service cannot claim the name, even though the registration has expired.

SRP provides a reasonably secure mechanism for publishing this information. Once published, DNS-SD clients using standard DNS lookups can readily discover these services.

### 4.2.2. Advertising Proxy

Hosts on infrastructure networks like Wi-Fi and Ethernet typically discover services using mDNS [RFC 6762]. Because Thread Devices offering those services may not be directly connected to the Adjacent Infrastructure Link and cannot directly receive or reply to link-local packets on that link, Thread Devices cannot directly use mDNS to answer those service discovery queries.

Instead, an Advertising Proxy acts on behalf of these Thread Devices. When a Thread Device registers its service(s) with the Service Registry on a Border Router, the Advertising Proxy capability on that Border Router makes those registered services visible on the Adjacent Infrastructure Link using traditional mDNS. This enables devices on the Thread Network to offer services and make them discoverable by devices using mDNS on an Adjacent Infrastructure Link without incurring the cost of sending multicast packets on the Thread Network.

### 4.2.3. Discovery Proxy

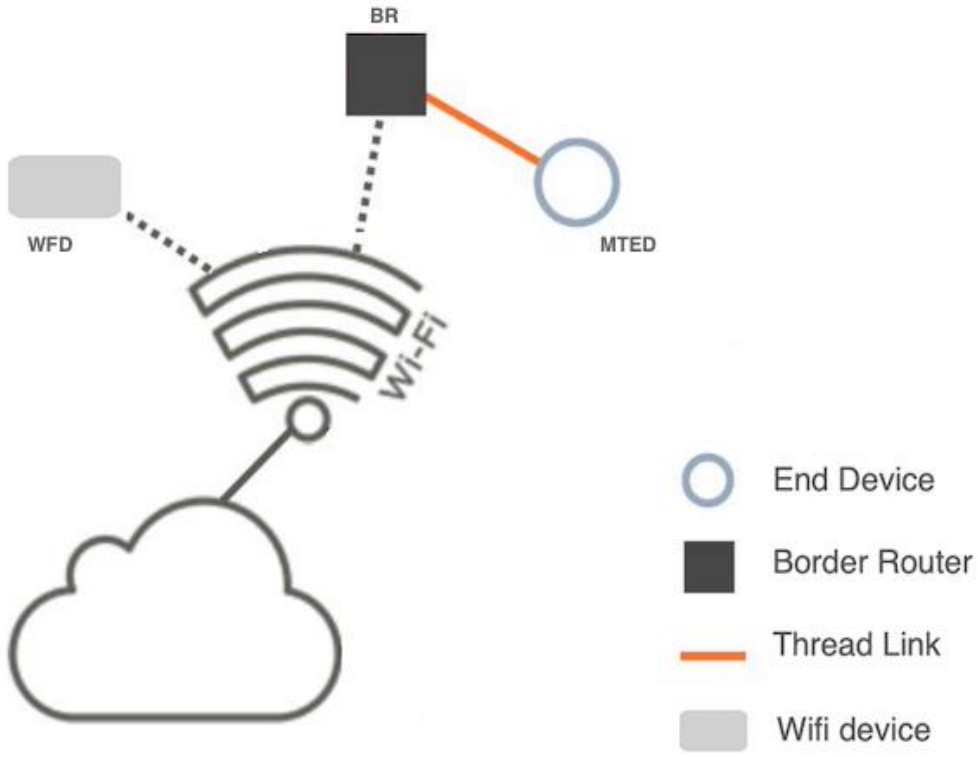
Hosts on infrastructure networks like Wi-Fi and Ethernet typically publish their services using mDNS. Because Thread Devices are not directly connected to the Adjacent Infrastructure Link and cannot directly send or receive link-local packets on that link, Thread Devices cannot directly use mDNS to discover those services.

Instead, a Discovery Proxy acts on behalf of the Thread Devices. A Thread Device communicates with the Discovery Proxy running on the Border Router and the Discovery Proxy performs mDNS queries on the adjacent infrastructure link on behalf of the Thread Device. This enables devices on the Thread Network to discover services available on the Adjacent Infrastructure Link without incurring the cost of sending multicast on the Thread Network.

### 4.2.4. Example Use Case: Matter service registration and discovery

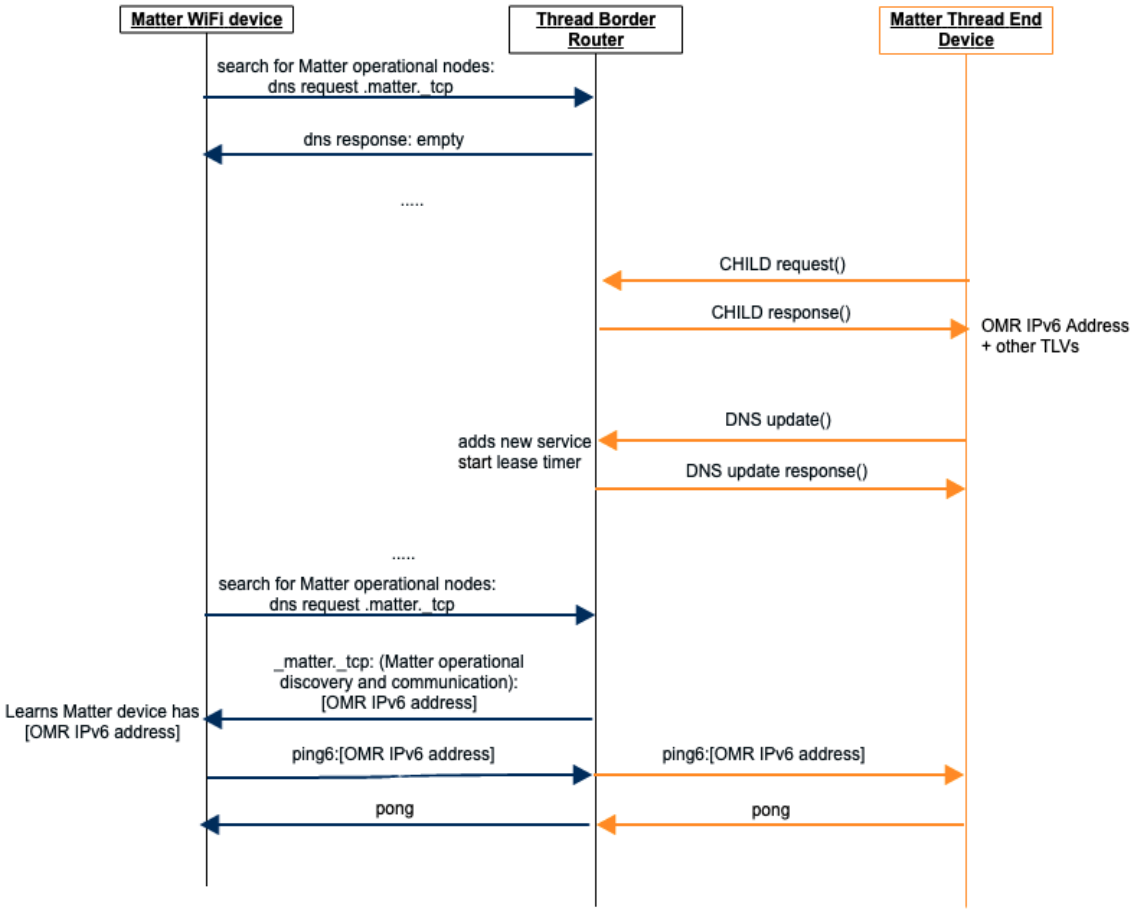
The following diagram shows a Thread 1.3.0 Border Router (BR) connected to a secured Wi-Fi network where a trusted Wi-Fi device (WFD) is monitoring available services using mDNS/DNS-SD.

A Matter Thread End Device (MTED) was commissioned to a Matter operational data (Fabric Id, Node Id) and Thread network credentials to join the secured Thread network.




During network attachment, the MTED receives IPv6 addresses, including the OMR address which will be used to send messages outside the Thread network.

The MTED starts its internal SRP client and recognizes the BR as the SRP server. This triggers the MTED SRP client to request registration of the Matter operational service. The BR will register that service and create a lease agreement, which means the MTED SRP client needs to report back periodically (on the basis of the SRP lease timer) to update the service in the BR SRP server tables.




The WFD periodically monitors the Wi-Fi network using mDNS queries searching for services named '_matter._tcp'. The BR Advertising Proxy proxy answers with the services available in the Thread network, including MTED operational service.

WFD learns the OMR address of the MTED. Now WFD can unicast over IPv6 to MTED. The BR will route the messages to/from Wi-Fi and Thread networks. This provides a communication path between these devices to create security sessions, pair application layer end points, and then reading cluster attributes and sending cluster commands.

The following picture shows the report from a mDNS Discovery app querying the local Wi-Fi network for all available services.


## 4.3. TCP

The implementation of Transport Control Protocol (TCP) as an endpoint is optional for a Thread Device, since none of Thread's management protocols make use of it. Hence, a Thread-compliant Device may implement the TCP initiator and/or listener role [RFC 793], [RFC 1122]. However, a Thread Certified Component must implement the TCP initiator and listener roles and must implement a well-defined API to use these roles. Implementers can then use this API to make use of TCP in a Thread Device they are developing.

TCP is well suited to efficient bulk transfer of data. Examples of bulk transfers include sending firmware updates to a Thread Device, and retrieving debugging log files from a Thread Device. The node sending the bulk transfer may be another node on the Thread Network, or a node outside the Thread Network communicating via a Thread Border Router.

TCP has specific advantages over UDP-based protocols. One clear advantage is that a TCP connection keeps a hole open in a network address translator (NAT) or firewall. Protocols that run over UDP can't provide this capability or

must provide traversal keep-alive traffic at excessively high rates to deal with short hole lifetimes.

Also, when TCP packets are sent using a maximum segment size (MSS) such that the TCP packet fits within a single Thread link-layer frame or a defined small number of Thread link-layer frames, TCP is more efficient than a UDP-based protocol. This is because UDP messages are sent in single IP packets, without regard to the underlying network's frame size. In order to send a UDP packet larger than a Thread 802.15.4 frame, the Thread 6LoWPAN adaptation layer is forced to fragment the UDP packet. The IP layer is not aware of this fragmentation, and so the UDP-based retransmission algorithm is forced to retransmit all frames in a fragmented message even when only one frame has been lost.

However, existing TCP implementations are not commonly tuned to work optimally over wireless mesh networks and within the smaller 802.15.4 frames. Therefore, Thread 1.3.0 specifies the elements and parameter values required for an efficient TCP implementation over Thread networks.

# 5. Summary

Building on the solid foundation of Thread 1.1/1.2, Thread 1.3.0 enables Matter, and simplifies development and deployment investments for device manufacturers by implementing features standardizing the Border Router role, simplifying firmware updates and increasing network robustness.