Thread 1.2 Base Features

June 2019

Thread 1.2 Base Features

DATE: June 2019

AUTHORS: Bruno Johnson, Alin Lazar, Ciaran Woodward

Table of Contents

Introduction

This white paper provides an overview of the Thread 1.2 Specification base feature set. These features are mandatory for all certified Thread 1.2 devices. The advanced features of Thread 1.2 devices and networks are fully interoperable with existing Thread 1.1 devices and networks, as the Thread Group is committed to seamless backward compatibility.

The base features of Thread 1.2 are designed to improve the scalability of Thread networks, by making them more responsive, and capable of a higher network density. New low power features further reduce the power consumption, channel utilization and communication latency of Sleepy End Devices. This enables even more efficient battery powered or energy-harvesting end products. New features for communication between multiple thread networks on the same LAN, and non-Thread devices on that LAN have been added.

These features improve integration with traditional networks and allow a new level of scalability for the commercial market by connecting multiple Thread Networks. A description of each of the base features is given below, along with the associated benefits for the user.

Additional white papers will be released in the future to detail the Thread Commercial Commissioning and Thread over Bluetooth LE extensions.

IEEE 802.15.4 Link Optimizations

The IEEE 802.15.4 Link Layer is the foundation for Thread communication, providing a Low Power, reliable wireless link to build upon. Thread 1.2 takes this further by applying several optimizations to improve its efficiency and performance. These link optimizations maximize battery life, improve responsiveness, and reduce network overheads for Sleepy End Devices (SEDs).

Battery life is improved by reducing the requirement for the SED to transmit network-specific control traffic. This means that the SED can spend less power transmitting, and more time sleeping, which has a dramatic effect on battery life.

Responsiveness is improved by making the SED more aware of when the parent has data that is pending for it. This allows the SED to retrieve messages more quickly than if it were required to explicitly request status updates with dedicated data requests.

Network density is improved by reducing the number of messages that must be sent over the air. This also improves the scalability of the Thread Network, allowing a greater number of Thread Devices to coexist in the same space.

These optimizations are particularly relevant for SEDs that expect to send application data frequently, but receive rarely, such as sensors. Sensors can take advantage of the frequent data transmissions to be responsive to incoming messages and keep the connection alive, without the overhead of transmitting regular data requests to the parent.

These benefits are enabled by two underlying optimizations - ‘Enhanced Frame Pending’ and ‘Enhanced Keep Alive’.

Enhanced Frame Pending

Enhanced Frame Pending (EFP) is a new feature in Thread 1.2 that improves the responsiveness and battery life of a SED. The EFP feature reduces the number of messages a SED must send over air, without compromising on behavior.

In the IEEE 802.15.4 standard, the only method that a SED can use to determine if its parent has data available is by explicitly polling for it by transmitting MAC Data Request command frames. This requires dedicated transmissions and additional network traffic.

Enhanced Frame Pending is a feature in Thread 1.2 that removes this transmission overhead. With EFP, any time a SED sends any data to its parent, the parent will report back the presence of any pending data to the SED in the Acknowledgement. This is a zero-overhead operation for the SED, and no additional octets are transmitted. EFP allows the SED to decide whether to sleep or extract data as shown in Figure 1.

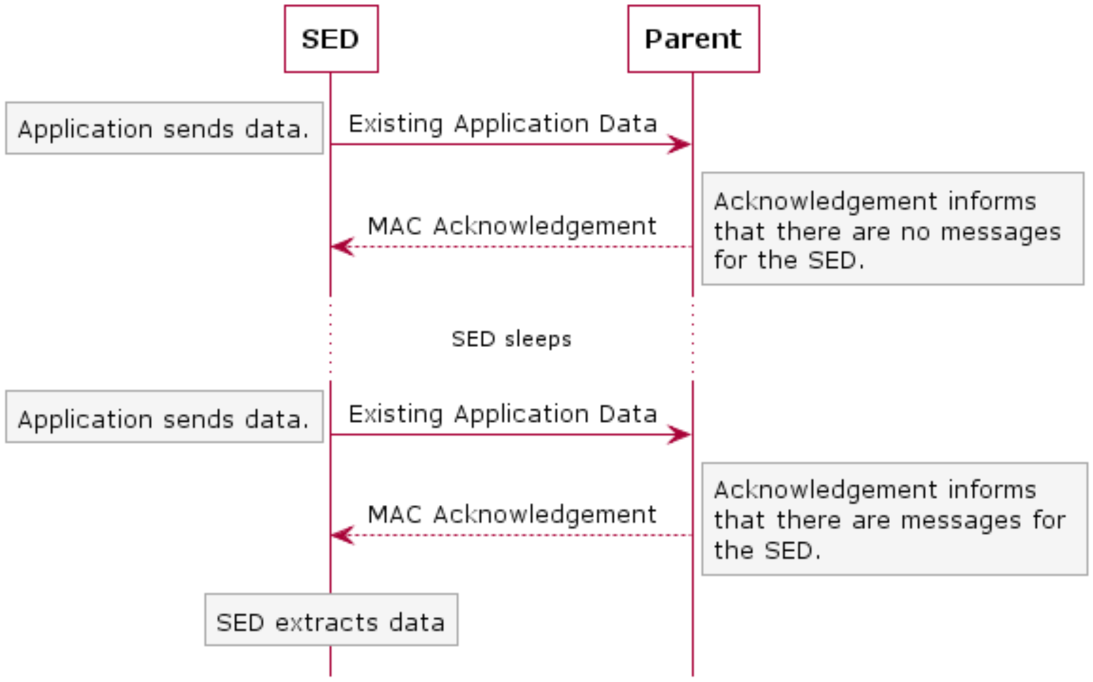
SED
Parent
Application sends data.
Existing Application Data
MAC Acknowledgement
Acknowledgement informs that there are no messages for the SED.
SED sleeps
Application sends data.
Existing Application Data
MAC Acknowledgement
Acknowledgement informs that there are messages for the SED.
SED extracts data

Figure 1 Enhanced Frame Pending

Enhanced Keep Alive

Enhanced Keep Alive is a new feature that reduces the amount of traffic that is required to be sent over the air by a SED to maintain a connection to the network. By allowing the SED to maintain its connection with any application data transmissions, it can dedicate its resources to application functionality.

In Thread 1.1, an explicit MAC Data Request transmission was required for SEDs to maintain a link to a parent. Enhanced Keep Alive is a feature that allows a SED to keep its parent link alive with any network transmission. This, in conjunction with EFP, means that nodes that receive data very rarely but transmit frequently will never have to send extraneous MAC Data Requests.

Synchronized Sleepy End Devices

Thread 1.2 adds an optional enhancement to SEDs, known as Synchronized Sleepy End Devices (SSEDs). SSEDs are highly responsive, low power end devices that can maintain a good quality of service in large, high density networks. This enables new use cases, such as a scalable network of low-latency, battery powered actuators in the commercial market.

SSEDs achieve this by using the Coordinated Sampled Listening (CSL) feature of IEEE 802.15.4-2015, maintaining synchronization with their parents and scheduling time to have their receivers enabled. The parent is responsible for buffering messages for the SSED until the SSED's receiver is enabled.

This enables low power end devices that have low link latency, as they can remain asleep for most of the time, but still wake up frequently for short periods in order to receive messages. Unlike a non-synchronized SED that must wake up regularly and transmit a MAC Data Request, SSEDs can simply wake up and listen for a short duration to determine if there are any buffered messages.

By reducing the number of over air messages that are required, more end devices can co-exist in the same space, as there is a lower chance of message collision. Continued synchronization for the link can occur on the back of normal application data exchange.

By removing the requirement for regular MAC Data Request transmissions, battery life is also improved.

SSEDs are particularly optimal for low power devices that primarily expect to receive with low latency, such as actuators. Actuators can take advantage of their regular receptions to keep the link synchronized and can listen with a frequency and duty cycle according to the application requirements. Large numbers of these low-latency actuators can exist in the same network without flooding it with MAC Data Requests.

Link Metrics

Thread 1.2 provides a set of features to allow a Thread Device to fully evaluate the links with its neighbors. This allows SEDs to optimize the link to their parent for the best performance. This is achieved by analyzing link quality while adjusting parameters such as transmit power, amplification or antenna diversity. These optimizations can be used to maximize battery life, solidify link reliability and improve network health. They are particularly relevant to SEDs and SSEDs, where transmit power consumption can be reduced, and large, dense networks, where the probability of transmission collision is higher.

Devices can execute a quick single-shot measurement of a link, or can continuously monitor the link quality, allowing for simple evaluation or powerful dynamic link optimization. The utility and flexibility of this feature allows any device to select the most suitable tools for the application.

Thread 1.2 provides three methods to extract link metrics data from a link. Firstly, a direct link metrics probe can be sent by a device, with an immediate response from the probing subject. Secondly, a link metrics series can be established between two nodes, where aggregated link metric data can be extracted in future. Finally, a link can be configured so that the probing subject includes link metrics data in the acknowledgement to any data transmission. This provides a full suite of tools that a device can use to optimize for the application.

The link metrics that can be extracted include the Link Budget, a count of messages, RSSI and IEEE 802.15.4 Link Quality Indicator.

Improving Network Health & Reducing Energy Consumption

By using the Link Metrics feature to reduce transmit power, the health of the entire network can be improved.

The IEEE 802.15.4 protocol uses the blind back-off CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) channel access mechanism. With CSMA/CA, a transmitting device first checks whether the channel is clear, before attempting transmission. If it observes that the channel is clear, it will start transmitting. If not, it will retry the entire procedure after a random time interval.

This approach works well when all nodes can “hear” each other, which is a rare case in large networks. If a node is not capable of discovering what is happening beyond its receiver range, then its transmission can lead to a collision, as the receiving node may be operating in the presence of an interferer. Known as the “hidden-node problem”, this problem affects most wireless communication protocols and, due to retries, can cause reduced throughput and increased latency and energy consumption.

Of course, improving receive sensitivity of a transmitting node improves the probability that it will be able to detect interference at a receiving node.

Similarly, reducing transmit power reduces the probability of any transmission causing collisions in parts of the network not intended for reception. This is where Link Metrics comes in. Not only does reducing transmit power save energy, but with large networks, it leads to a healthier network.

Scalable Heterogeneous Networks

Connected IoT devices in home and buildings have widely different requirements that determine their data processing, power consumption, and connectivity characteristics. Some appliances may need to be mains-powered and have a high-bandwidth connection, while some sensors and actuators can be battery operated and require lower-bandwidth but more flexible wireless connections.

Thread 1.2 introduces protocol enhancements allowing to easily interconnect IoT devices with such diverse characteristics either to each other or to local or remote servers. With these enhancements, constrained wireless low-power IoT environments are seamlessly integrated within large scale installations spanning commercial office buildings, multi-dwelling units, or even campus networks.

Thread 1.2 also describes how multiple wireless Thread Networks or Partitions can be interconnected by larger bandwidth network segments such as those based on Ethernet or Wi-Fi, and how devices at the border of the Thread Network help to manage IP forwarding across subnet boundaries in a standardized fashion.

To achieve that, Thread 1.2 continues to rely on the solid foundation provided by end-to-end IP/IPv6 technologies. These technologies have broad field deployment which allows easy integration with existing network infrastructure for Cloud or Edge processing. Furthermore, these technologies and management infrastructure are already familiar to IT administrators.

Figure 2 shows an example of a multi-segment network using a backbone link which interfaces multiple Thread Networks or Thread Network Partitions with other local or cloud IPv6 infrastructure.

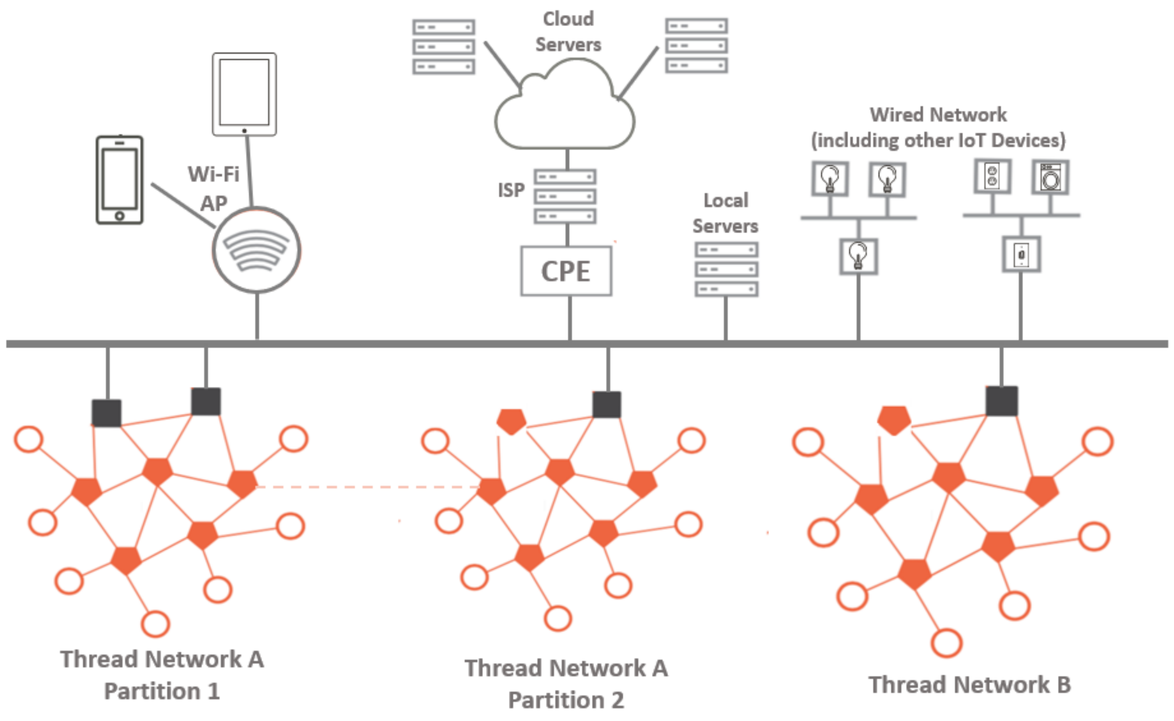
Cloud Servers
Wi-Fi AP
ISP
CPE
Local Servers
Wired Network (including other IoT Devices)
Thread Network A Partition 1
Thread Network A Partition 2
Thread Network B

Figure 2 Multiple Thread Networks and Partitions interconnected using a backbone link

Thread 1.2 introduces the following significant scalability enhancements:

- Definition of a Backbone Router function for Thread Border Routers. A Backbone Router facilitates intercommunication outside the Thread Network with a backbone network segment.
- Introduction of the concept of Thread Domains. Thread Domains greatly enhance the flexibility of managing address and network access configuration of Thread Devices in multiple Thread Networks in a coordinated way.
- Additional Thread Device and Backbone Router protocol definition that facilitates multicast and unicast IPv6 packet forwarding across the boundaries of Thread Networks.

Backbone Routers

A Backbone Router (BBR) is a function that resides in a Thread Border Router that is used to:

- Facilitate the flow of inbound and outbound multicast IPv6 packets with a scope larger than the realm-local wireless mesh network to and from a Thread Network Partition.
- Keep track of Thread Devices within shared Thread Domain configurations, also jointly with other BBRs.

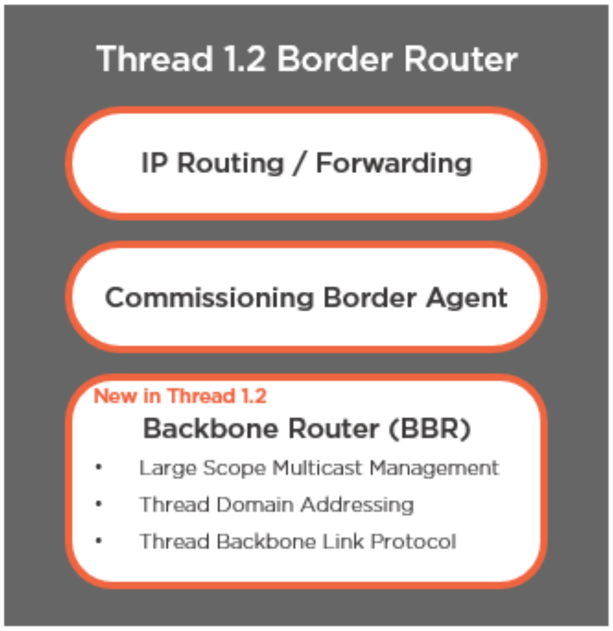

Figure 3 Thread 1.2 Border Router functionality overview including Backbone Router (BBR) function

Figure 3 shows an overview of the Thread 1.2 Border Router functionality.

Within the Thread Network or subnet, the BBR advertises specific server data to describe its Backbone Router functionality to other nodes on the Thread network it services.

When fulfilling the BBR function, a Border Router can synchronize information and operation with other BBRs. If BBRs share external interfaces to the same non-Thread Network segment (such as a home LAN, a building managed network or VLAN), then the BBR synchronization is performed using this external network as a Backbone Link by implementing the newly defined Thread Backbone Link Protocol.

As already described by the Thread 1.1 Specification, a Thread Network may have multiple active Border Routers. This has the advantage of providing redundancy, resilience, and prevents a single point of failure for external communication. As a result, the BBR function may also be available on several Border Routers attached to the same Thread network. To allow for features coordination, the BBR function operates in one of two operational states: Primary or Secondary. By default, devices start in Secondary BBR state then multiple BBRs participate in an election algorithm to select a Primary BBR.

A Primary BBR keeps a list of all Domain Unicast Addresses (DUAs) registered by Thread Domain Devices in its Thread Network. This enables the Primary BBR to answer IPv6 Neighbor Discovery queries on the Backbone Link on behalf of the registered Thread Domain Devices. Answering these queries is what makes a Thread Domain Device reachable by IPv6 nodes in an external network.

Any Secondary BBR is on standby and can potentially take over the Primary BBR role if the current Primary BBR stops operating.

For networks with a configured Thread Domain IPv6 Prefix, the BBR provides routing and forwarding for packets having Thread Domain Address destinations which may be initiated by external (non-Thread) Devices. On the Backbone Link, this function is achieved using the Thread Backbone Link Protocol and Neighbor Discovery (ND) Proxy operation.

Coordinated Large Scale Multicast

In a building, campus network, or even a large residential network, an input such as a lighting switch change needs to simultaneously send an action to multiple luminaires at large scale and with good time synchronicity. Devices publishing sensor data that may be of interest to multiple consumers also use multicast transmissions (for example to disseminate critical alerts).

Propagating multicast across a heterogeneous network consisting of multiple segments or subnets using end-to-end standardized IP multicast reduces complexity and allows flexible configurations that do not depend on the specifics of link layer segments. This allows modular, de-coupled network configuration. To achieve larger scale multicast operation, Thread 1.2 hosts can subscribe to IPv6 multicast groups for multicast scopes higher than link-local.

In larger installations, an IPv6 network may be subdivided into multiple IPv6 subnets. Each of these subnets may use a distinct network technology (for example Thread, Ethernet, or Wi-Fi). In such installations, there is often a need to address groups of IPv6 devices using IPv6 multicast communication in a uniform manner, without requiring knowledge of the network topology on the application level (for example, without knowing where in the network topology the group members are connected or what network technology or intermediate routers are used to connect them). Any of the IPv6 nodes are potential producers of IPv6 multicast messages or consumers listening to an application-specific IPv6 multicast address.

Thread 1.2 defines management functions for a Thread Border Router and registration functions for other Thread Devices to enable seamless IPv6 multicast over multiple IPv6 subnets. What is enabled includes:

- Thread Devices receiving multicast packets coming from a source outside its Thread Network.
- Thread Devices sending multicast packets that are forwarded beyond its Thread Network.
- Forwarding of multicast packets coming from one Thread Network to another Thread Network without duplication.

The main approach taken is to use the concepts defined by the Multicast Listener Discovery v2 (MLDv2) protocol defined by IETF RFC 3810 while avoiding the complexity and traffic overhead on the Thread Networks that would result from direct usage of the MLDv2 protocol in a constrained wireless mesh. Functions on the Backbone Router are defined such that if at least one Thread Device on the Thread Network is registered at the BBR as a multicast listener for a specific multicast group, this multicast traffic is forwarded by the BBR onto the Thread Network using Multicast Protocol for Low Power and Lossy Networks (MPL) defined by IETF RFC 7731. MPL provides an optimized way of propagating multicast traffic to nodes which is superior to simply flooding network segments with multicast traffic.

BBRs use flexible configured propagation features to allow propagating multicast traffic across the network border depending on traffic direction (inbound or outbound) and IPv6 multicast scope.

Backbone Routers implement the MLDv2 protocol directly on its Backbone Link to facilitate integration with standard IPv6 (multicast) routers or MLD-snooping switches. A BBR will use MLDv2 on its External Interface to communicate to the wider IPv6 LAN/WAN about IPv6 multicast groups it needs to listen to, on behalf of the multicast listeners on its local Thread Network. The groups it listens to are those in its Multicast Listeners Table. MLDv2 enables IPv6 multicast packets to be received by Thread Devices even if these packets need to come through one or more upstream IPv6 multicast routers. For example, an IPv6 multicast Router on the Backbone Link will listen to the MLDv2 Multicast Listener Reports sent by a BBR and based on these reports, adapt its multicast routing topology.

By default, a Primary BBR forwards IPv6 multicast packets of admin-local or higher scope, that it receives on its Thread Interface, onto the Backbone Link. Non-Thread IPv6 Routers on the Backbone Link that support multicast routing protocols may then further route these IPv6 multicast packets to other network segments (if required). Another Primary BBR for a different Thread Network on the Backbone Link may forward these packets onto its Thread Network using MPL if a listener for these packets exists on its Network. BBR multicast packet filtering rules are also adhered to, aiming to reduce the load of unneeded multicast traffic on the Thread Networks.

Figure 4 and *Table 1* are providing an overview of the multicast propagation across the Thread Network border.

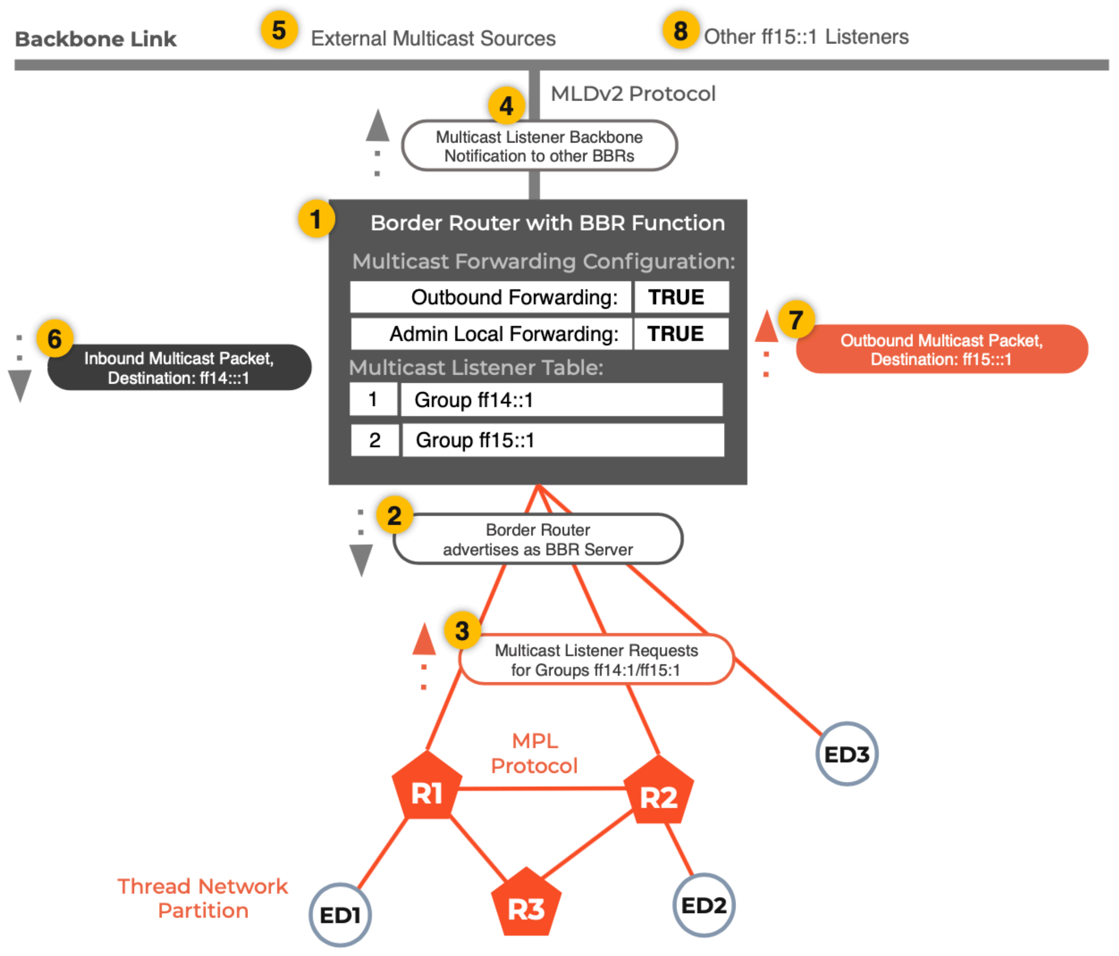
Backbone Link
5 External Multicast Sources
8 Other ff15::1 Listeners
MLDv2 Protocol
4 Multicast Listener Backbone Notification to other BBRs
1 Border Router with BBR Function
Multicast Forwarding Configuration:
Outbound Forwarding: TRUE
Admin Local Forwarding: TRUE
Multicast Listener Table:
1 Group ff14::1
2 Group ff15::1
6 Inbound Multicast Packet, Destination: ff14:::1
7 Outbound Multicast Packet, Destination: ff15:::1
2 Border Router advertises as BBR Server
3 Multicast Listener Requests for Groups ff14:1/ff15:1
MPL Protocol
R1
R2
R3
ED1
ED2
ED3
Thread Network Partition

Figure 4 Multicast forwarding facilitated by BBR function

1	A Thread Border Router with a BBR function is started and optionally configured administratively for forwarding multicast with scope larger than realm-local across the Thread Network border.
2	The Border Router advertises the BBR function as a BBR Server to the Thread Network Partition.
3	Thread Devices becoming aware of the BBR function on the Border Router register as IPv6 multicast address listeners (for higher than realm-local scopes) to the BBR which adds the registration to the BBR Multicast Listener Table.

4	The BBR operates on the Backbone Link by implementing the MLDv2 protocol with other devices on the Backbone Link on behalf of the registered Thread devices. It also advertises the Thread Devices multicast group registrations to other BBRs participating to the Backbone Link.
5	External devices to the Thread Network Partition generate multicast data which reaches the Border Router on the Backbone Link.
6	On the Border Router, the inbound multicast packets are filtered based on whether there are interested listeners on the Thread Network and BBR administrative Multicast Forwarding Configuration. Inbound multicast packets passing the filtering are propagated to the wireless mesh Thread Network using the MPL Protocol.
7	Thread Devices can also generate multicast traffic to destinations with a larger scope than realm-local which may be of interest to IPv6 devices outside the Thread Network. The outbound multicast packets are propagated within the Thread Network to the Border Router which applies outbound filtering rules and based on participation to MLDv2 protocol devices on whether to forward the packets on the Backbone Link.
8	Devices outside the local network partition (which may be other Thread devices interconnected via other Border Routers) receive the outbound multicast packets.

Table 1 Multicast forwarding facilitated by BBR function (Figure 4 steps)

Thread Domain Unicast Addressing

A Thread Domain is a set of Thread Devices that receive and apply common Thread Domain operational configuration. The Thread Domain operational configuration enables Thread Devices to join and participate in larger interconnected systems extending beyond the limits of a single Thread Network. A user or network administrator may use functions of Thread Commissioning or of Thread Border Routers to set up a common Thread Domain operational configuration for Thread Devices belonging to different Thread Networks or Partitions having potentially different per-network credentials. The BBR function that resides in a Thread Border Router performs coordinating tasks jointly with other Backbone Routers that are part of the same Thread Domain.

The main attribute of a Thread Domain is the Thread Domain Prefix, which is an IPv6 address prefix that allows Thread Devices to configure Domain Unicast Addresses (DUAs).

A Thread Domain can seamlessly integrate multiple Thread Networks and non-Thread IPv6 networks. One main benefit of the Thread Domain networks is that devices can join any available Thread Network configured with a common Thread Domain which reduces the need for manual network planning or costly manual reconfigurations when network size or data volume are scaled up.

A set of synchronized BBRs interconnecting multiple Thread Networks can ensure that Domain Addresses can be found and routed across networks, thus extending the Address Query mechanism from a single Thread Network to multiple Thread Networks, Thread Network Partitions, or network segments sharing a common Thread Domain configuration.

To provide IPv6 connectivity to DUAs on the Backbone Link, Primary BBRs implements IPv6 Neighbor Discovery (ND) Proxy based on IETF RFC 4389. A Primary BBR acts as a selective proxy only for registered Domain Addresses by devices in its Thread Network Partition. This enables a regular IPv6 Router or another BBR connected to the Backbone Link, to interact with the Thread Domain Device using the ND protocol as if it were a neighbor on the same IPv6 link.

The Thread Domain Prefix is indicated as an on-link prefix for the Backbone Link. This enables a BBR to process IPv6 ND queries for Thread Devices received on the Backbone Link in a standard way using IPv6 ND Proxy functions.

Figure 5 and Table 2 are providing an overview of the Thread Domain Configuration for multiple Thread Networks or Partitions and the forwarding of IPv6 packets having Domain Unicast Address destinations.

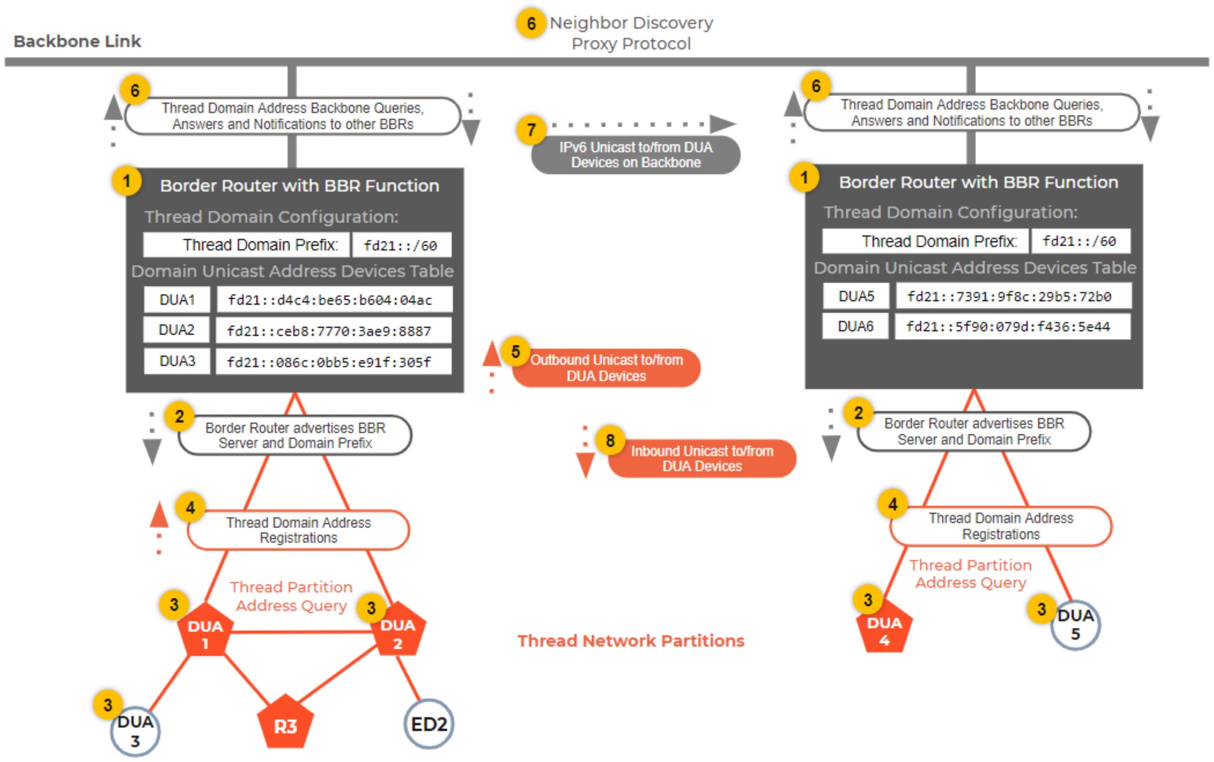
Backbone Link
6 Neighbor Discovery Proxy Protocol
6 Thread Domain Address Backbone Queries, Answers and Notifications to other BBRs
7 IPv6 Unicast to/from DUA Devices on Backbone
6 Thread Domain Address Backbone Queries, Answers and Notifications to other BBRs
1 Border Router with BBR Function
Thread Domain Configuration:
Thread Domain Prefix: fd21::/60
Domain Unicast Address Devices Table
DUA1 fd21::d4c4:be65:b604:04ac
DUA2 fd21::ceb8:7770:3ae9:8887
DUA3 fd21::086c:0bb5:e91f:305f
1 Border Router with BBR Function
Thread Domain Configuration:
Thread Domain Prefix: fd21::/60
Domain Unicast Address Devices Table
DUA5 fd21::7391:9f8c:29b5:72b0
DUA6 fd21::5f90:079d:f436:5e44
5 Outbound Unicast to/from DUA Devices
8 Inbound Unicast to/from DUA Devices
2 Border Router advertises BBR Server and Domain Prefix
4 Thread Domain Address Registrations
Thread Partition Address Query
3 DUA 1
3 DUA 2
3 DUA 3
R3
ED2
Thread Network Partitions
2 Border Router advertises BBR Server and Domain Prefix
4 Thread Domain Address Registrations
Thread Partition Address Query
3 DUA 4
3 DUA 5

Figure 5 Thread Domain Configuration and Domain Unicast Addressing forwarding

1	One or multiple Thread Border Routers with BBR functions are started and configured administratively for managing Thread Domains. If the same Thread Domain configuration is performed on multiple Border Routers, the same Thread Domain Prefix is configured for each BBR.
2	The Border Router advertises the BBR function as a BBR Server to the Thread Network Partition along with the Thread Domain Prefix.
3	Thread Devices becoming aware of the BBR function on the Border Router configure Thread Domain Unicast Addresses (DUAs) based on the Thread Domain Prefix.
4	Thread devices register their DUA addresses to the BBR which adds the registrations to the BBR Domain Unicast Addresses Devices Table. Legacy Thread 1.1 devices may not register such addresses (e.g., R3, ED2 in Figure 5).
5	DUA Devices on a Thread Network Partitions initiate unicast traffic to another DUA Device destination that may or may not be attached to the

	same Thread Network partition. Address queries for a DUA destination reaching the local partition BBR allow the BBR to determine if the destination is external to the Thread Network Partition.
6	The BBR initiates queries on the Backbone Link to other BBRs and uses the Neighbor Discovery Proxy Protocol to determine whether to forward the unicast packet with the DUA destination to another BBR.
7	If the queries for the destination are resolved to determine that the DUA destination packets are to be forwarded to another BBR, the first BBR sends the IPv6 on the Backbone Link.
8	The receiving BBR attached to a different partition forwards the unicast packets forwards the data to the DUA Device destination.

Table 2 Thread Domain Configuration and Domain Unicast Addressing forwarding (Figure 5 steps)