

MeshDriver - Technical Overview

Abstract

Overview of the mesh routing, control and signaling protocols utilized in MeshDriver, software architecture and integration considerations.

Copyright © 2004-2009 Meshcom Technologies, Inc. All rights reserved. Meshcom and Meshcom MeshDriver are registered trademarks of Meshcom Technologies Inc.

MeshDriver is available for downloading at EmbedOne website <http://embedone.com>

Table of Contents

1. MeshDriver Overview	3
2. Benefits.....	3
3. Meshcom Mesh Routing in Practice.....	4
3.1 Neighbor and Route Discovery	4
3.2 Mobile Nodes.....	5
3.3 Node Failure.....	6
4. Meshcom Mesh Protocol	7
4.1 Meshcom Mesh Protocol (MMP) Overview	7
4.2 Node Identification	7
4.3 Fragmentation	7
4.4 Superpacketing.....	8
4.5 Handshake, Authentication and Encryption.....	8
4.6 Control Data Units (CDU).....	8
4.7 Beacons (HELLO) and Links.....	9
4.8 Routing Control Data Units and Route Selection	9
4.9 Ethernet master (ETHMASTER) Functionality	10
4.10 Routed Data Units (RDU)	10
5. MeshDriver Product Keys	10
5.1 Types of Product Keys	Error! Bookmark not defined.
5.2 Information Contained in Product Keys	Error! Bookmark not defined.
Appendix 1. Integration of Meshcom MeshDriver OEM.....	11
1.1 Wireless LAN Operating Modes.....	11
1.2 Multi-hop Networks, Multi-Radio Systems.....	11
1.3 Software Integration	11
1.4 Required Interfaces from Operating System	12
Appendix 2. Glossary and Acronyms	13

1. MeshDriver Overview

MeshDriver creates a virtual network interface (“mesh interface”) by multiplexing and de-multiplexing L2 traffic from one or more hardware ports. A MAC address is assigned to the mesh interface by MeshDriver. The assigned MAC address of the mesh interface serves as a node ID and must be unique in the scope of Mesh network.

All the MeshDriver-enabled devices (“nodes”) in the mesh network can be thought of as being connected to the same Ethernet segment regardless of number of hops between them. Similar to conventional bridge MeshDriver joins several Ethernet segments, but unlike conventional bridge it makes intelligent routing decisions concerning packets that should go several hops to destination.

The easiest way to understand how MeshDriver-enabled network functions is to regard it as a distributed Ethernet switch, where each node does its part of packet routing. When a node joins a mesh network it is equivalent of being plugged in that switch.

For communication with other nodes MeshDriver uses proprietary Layer 2 Meshcom Mesh Protocol (MMP) of type 0x88ed. It performs routing by encapsulating (not MeshDriver-enabled) end-user L2 frames in MMP frames which it passes through the links to neighbor nodes. To establish links with other nodes MeshDriver broadcasts and listens to beacons on controlled interfaces. Thus it relies on pre-established physical connectivity between network ports, such as Ethernet cables or WLAN links. Managing physical connectivity between the nodes is beyond MeshDriver capabilities.

2. Benefits

Current routing protocols used in modern IP networks are not very suitable for use in dynamic wireless networks. Most of the protocols available today assume a fairly static connection between two network elements in a network, and base their routing decisions on a static configuration or the number of routers between the source and destination alone. MeshDriver constantly measures the link connections between two nodes and can take immediate action when changes occur.

Another difference is that in the existing protocols, all of the routing information is usually transmitted to each of the routers in a network, but in a fairly infrequent rhythm. Especially mobile networks require very fast pace of updates since the routes change rapidly, which would mean unacceptably large consumption of bandwidth if traditional routing protocols were to be used. MeshDriver has very optimized bandwidth usage even though the update frequency is high, by transferring only the data needed and when needed.

Configuration of a MeshDriver-enabled network is also very easy compared to these routing protocols, since MeshDriver operates on a lower level (L2) and combines several network ports into a single network interface.

Protocol	OSPF	AODV (RFC 3561)	RIP	Static Routing	MeshDriver
Type	Proactive	Reactive	Proactive	Proactive	Reactive
Routing based on	hop count manual settings	hop count	hop count manual settings	manual settings	hop count error rate custom metrics
Operating Layer	Layer 3	Layer 3	Layer 3	Layer 3	Layer 2
Network can react to changes in topology	Yes	Yes	Yes	No	Yes
Easy expansion of network	No	Yes	No	No	Yes
Automatic configuration	No	Yes	No	No	Yes
Topology change convergence speed	Average	Good	Average	None	Excellent
Routing data overhead	Large	Small	Average	None	Small

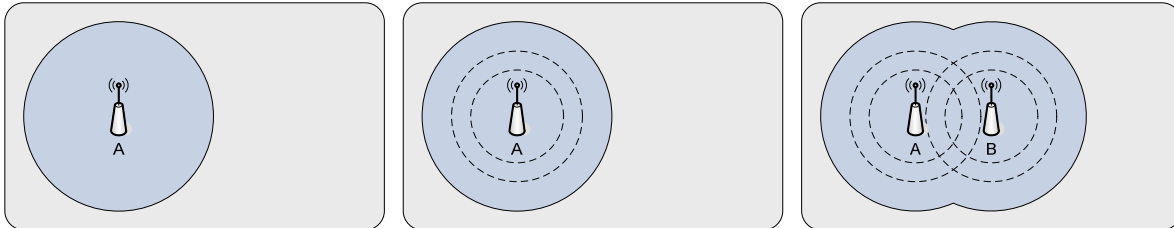
3. MeshDriver Mesh Routing in Practice

The operation of MeshDriver Mesh Routing is illustrated by three different use cases.

3.1 Neighbor and Route Discovery

The following diagrams illustrate how basic processes of finding other nodes in the network.

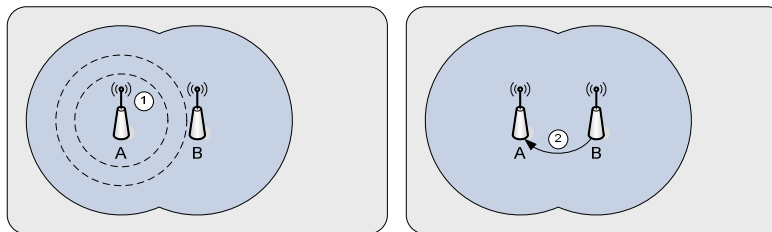
Neighbor Discovery



Each mesh network formation begins with the first node to boot up (left picture) and start sending HELLO beacons (center picture).

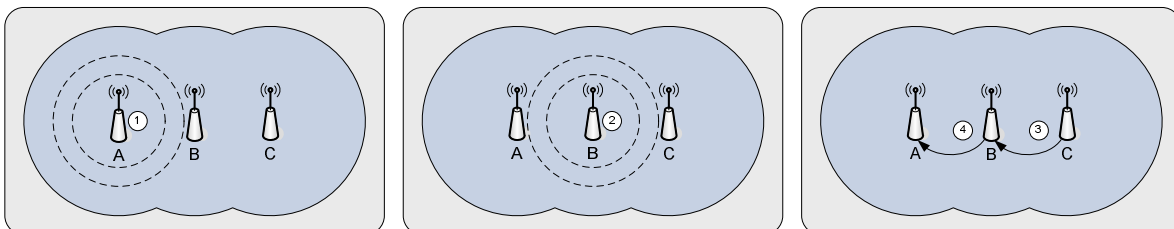
When another node arrives to the network or is powered on, it also starts to send HELLO beacons and the nodes recognize the presence of each other (right picture). When the nodes receive HELLO beacon from each other, they initiate the handshaking process. After the handshaking and authentication, they have formed links to each other and can start actual communication.

Route Discovery with Two Nodes



When node A has some data to send to node B, it initiates a route discovery process by sending route request (RREQ) to all nodes around it (1). Since the route request is broadcasted by node A to everyone, it is also heard by node B. Node B checks whether the destination in request is the node itself or otherwise known. Since the requested route is to node B itself, node B sends a route reply (RREP) to node A (2), stating that the destination can be reached through node B.

Route Discovery with Three Nodes



If a third node, node C, is present in the network and the network is organized so that node A can reach node B but not node C, and node B can reach both of the nodes, the route from A to C must go through node B. In that scenario, the following steps occur when A wishes to send data to node C:

1. Node A broadcasts a route request (RREQ)
2. Node B hears the broadcast, checks its routing table and since it does not have a route to node C, it broadcasts the request again

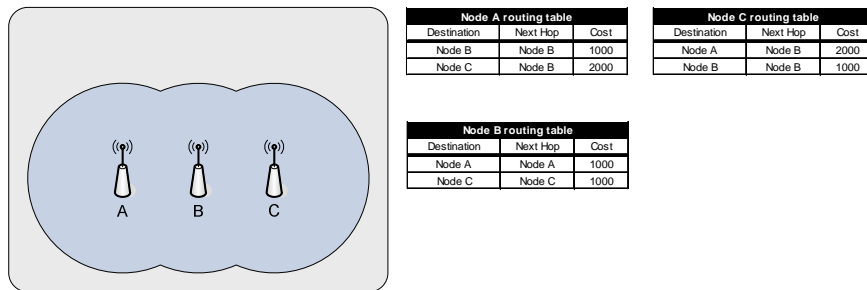
3. Node C hears the request broadcasted by node B and sends a route reply (RREP) to node B, with the information that the original node to request the route was node A.
4. Node B receives the route reply sent by node C, checks the original requestor and sends it to node A.

In step 2, if node B would have known the route to node C, it could have responded immediately on behalf of node C, without having to broadcast the message at all.

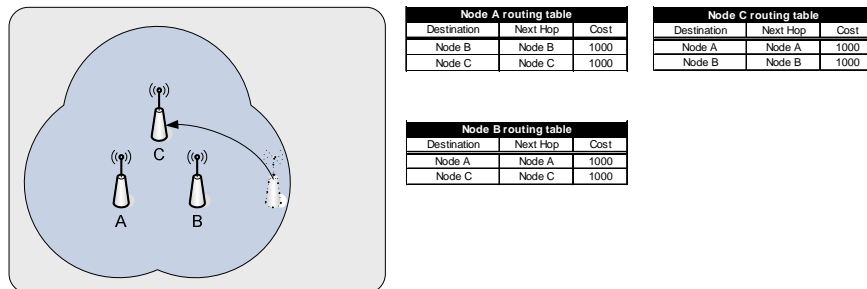
Along with the route request (RREQ), also the cost of the route is recorded and then sent back in route reply (RREP). With many neighbors, it is possible and even likely that a node requesting a route receives several route replies which each represent alternative routes to the requested destination. The route with least cost is kept and recorded into the routing table.

3.2 Mobile Nodes

Continuing from the previous example (three nodes that have been initialized and they have requested routes to each other), the internals of a node's decision process can be examined deeper. In the figure below, the nodes have been labeled "A", "B" and "C".

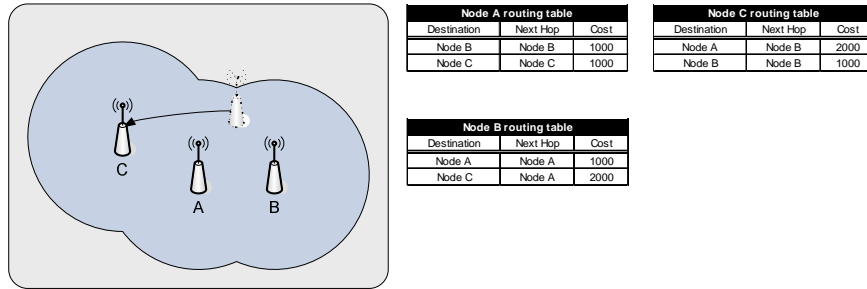


On the right there are also routing tables of each of the nodes – node A can connect to node C only through node B (two hops), but node B can communicate directly with both A and C (one hop). Assuming that the nodes A and C keep communicating with each other, the routes are also refreshed once in a while (MeshDriver default is 15 seconds).



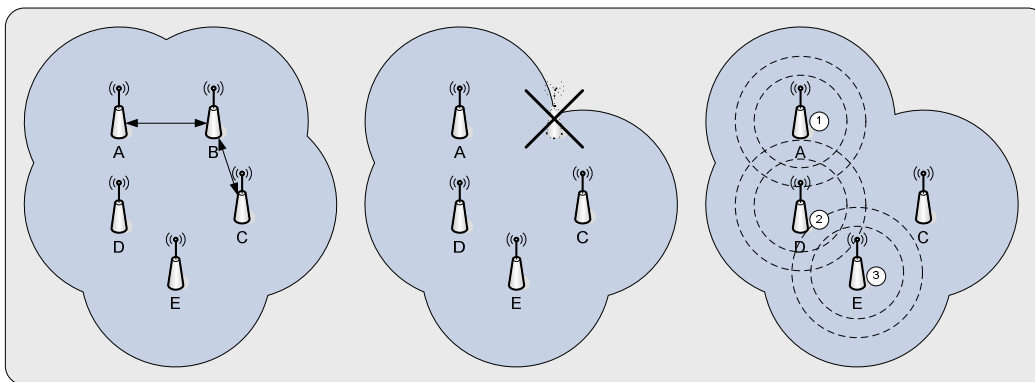
If node C moves from the vicinity of node B close to both nodes A and B, node A first continues sending data through node B as it is used to. When the periodic refreshing is triggered, node A broadcasts a new route request (RREQ) to the network. In response to the request, node A will have two responses; one from node B stating cost 2000 and other from node C stating cost 1000. Since the response from node C has less cost, it is the preferred route and will be recorded to the routing table.

Note that before the periodic refreshing is done, the communication between nodes A and C still works, but since it is done through node B it is not the most efficient way.



If Node C continues its movement and proceeds away from the range of Node B, node B removes the direct route it had when the link is expired due to lack of hello beacons. It then starts a new route discovery by broadcasting a route request (RREQ), which will be responded to by the intermediate node A, who already knows a route to node C.

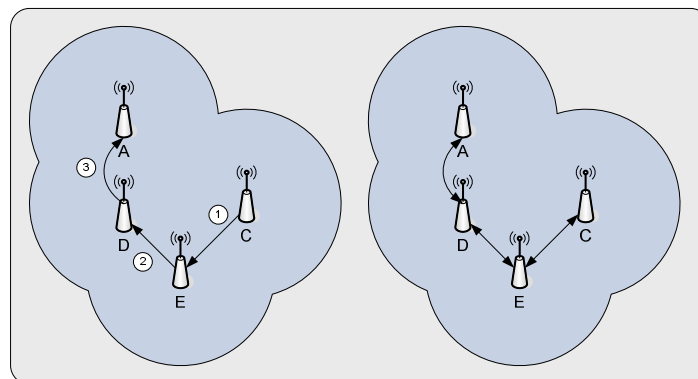
3.3 Node Failure



In a network consisting of five nodes (A, B, C, D, E), a connection between two nodes that are exchanging data (nodes A and C) can break (by node B being powered down or moving out of range, for example).

When node A loses the link to a next hop it has to a destination (B is the next hop for destination C), it removes the route to destination (node C).

When traffic should be sent from node A to node C again, node A has to search for a new route because the previous one was removed. Node A broadcasts (1) the route request (RREQ), which is then re-broadcasted by nodes D (2) and E (3), until the request reaches destination (node C). While the route request (originally sent by node A) is propagated through the network, the nodes that hear it (D, E and finally C) will also get the knowledge of a route to the original sender (node A)



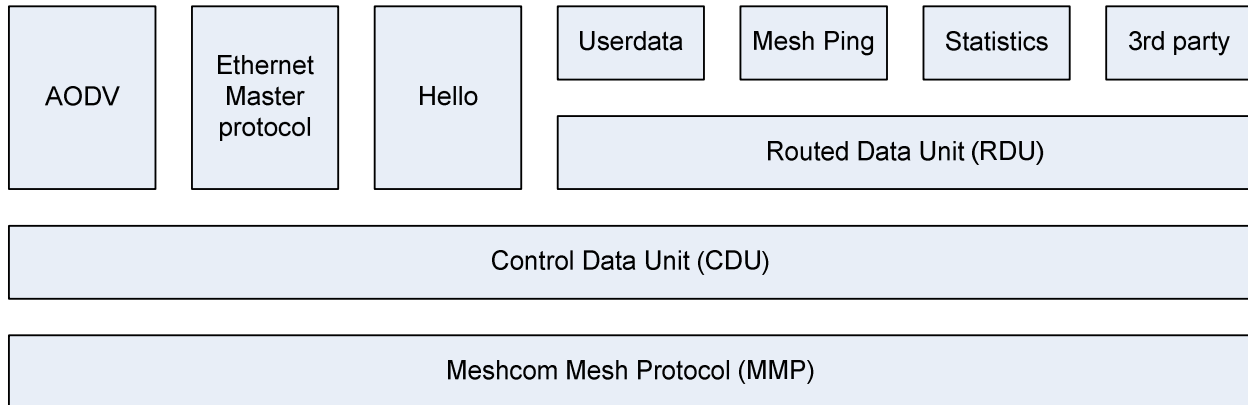
After node C receives the request, it will respond to the request by sending unicast response through the best known route to the original node that sent the request (node A). Often this route has been learnt through the request, but it is also possible that the node has a route to the original sender by

some other means already. The reply is forwarded by the intermediate nodes according to their routing table (1, 2, 3), which in this scenario was also updated by the original request.

After the reply has reached the node sending the original request, both ends (node A and C) have gained the route to each other and can proceed sending the actual data.

4. Meshcom Mesh Protocol

Meshcom Mesh Routing is based on a suite of protocols developed for mesh communications. All nodes in the mesh network communicate using Meshcom Mesh Protocol (MMP). MMP is used to transfer control information such as routing information, link keep-alive beacons as well as the actual data that is delivered through the network.



Overview of Meshcom Mesh Routing protocols

4.1 Meshcom Mesh Protocol (MMP) Overview

MMP protocol is used on top of the actual physical link-layer protocol (typically 802.3 Ethernet or 802.11 Wireless LAN). The MMP frames are distinguished by the IEEE Ethertype 0x88ed.

MMP protocol manages link-level communication with the following features:

- Destination & source identification (See section 4.2)
- Fragmentation (See section 4.3)
- Superpacketing (See section 4.4)
- Message authentication and encryption (See section 4.5)

MMP protocol transfers data in units called Control Data Units (CDU) as described in section 4.6. One MMP transmission (individual frame or packet) can include several CDUs or just a part of a large CDU.

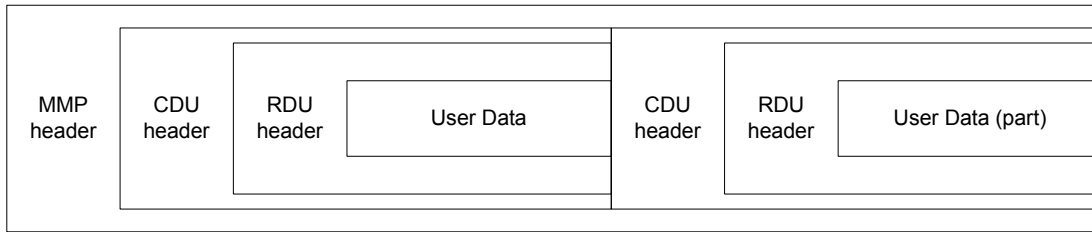
4.2 Node Identification

All nodes in a mesh network are identified by unique 48-bit node identifiers that are similar to the physical MAC addresses. This node identifier is used also as the mesh node's mesh MAC address. Each of the data units sent from one node to another includes the address of recipient (destination node) and sender (source node). Node identifiers are used in both link-level (from neighbor to another) as well as end-to-end (routed through a mesh network) communication.

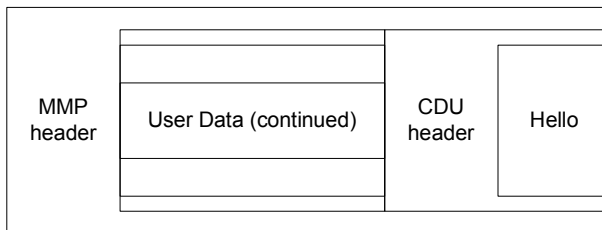
4.3 Fragmentation

MMP allows transmission of up to 4 kilobyte long control data units (CDUs). Since some modern packet switched networks (such as Ethernet or Wireless LAN) support a smaller amount of bytes to be transferred at a time, MMP also includes support for fragmentation. Fragmentation can be combined with superpacketing, resulting in MMP frames that may have several CDUs of which one is only partly transferred.

An example transmission of a fragmented CDU in two MMP frames:



MMP frame with one whole user data frame and a fragmented second user data frame



Second MMP frame with rest of the fragmented user data frame and a hello CDU

4.4 Superpacketing

One MMP frame can contain several CDUs of same or mixed types. Most often CDUs that are sent by the same protocol are combined to one MMP frame (such as routing updates).

4.5 Handshake, Authentication and Encryption

Whenever a node notices a new mesh neighbor, it starts a handshake process. During the handshaking process, the nodes agree on an authentication mechanism to be used. If one of the nodes requires an authentication method the other does not support or the node does not have the required credentials, the authentication is terminated. Nodes also negotiate encryption keys during the process.

Link can have the following handshake/authentication states:

- Handshaking (initial state, when HELLO beacon has been received and the handshake is in process)
- Authenticating (handshake is done and agreed authentication in process)
- Unauthenticated (authentication did not succeed, link will not be used)
- Authenticated (authentication successful and link can be used)

Since MeshDriver semantics is very close to WiFi, MeshDriver security model was taken from 802.11i standard. When the two nodes establish a link they first decide which one will take a role of an Access Point (AP), and which one will act as a Workstation (WS) in 802.11i semantics. After that mutual authentication happens (if required by nodes' authentication policy) which derives Pairwise Master Key (PMK) for that session. Following 802.11i standard PMK is used to negotiate Pairwise Transient Key (PTK) by AP. All communication through the link is encrypted using CCMP. Broadcasts are currently not encrypted. Authentication policy can be specified for the whole node or for particular port.

Policy code	Description
Open	No authentication required. In that case PMK is not derived and no encryption is done.
SRP	Secure Remote Password protocol for preshared key/password authentication (RFC 2945). Its purpose is similar to WPA PSK. The protocol was trivially modified to allow mutual authentication and decrease the computational burden – without considerable security compromise.

4.6 Control Data Units (CDU)

Control Data Units are transferred from a neighboring node to another using MMP. CDU is a simple container for several types of messages that are used to either manage the network or transfer data. Each CDU contains headers that specify the CDU type and the length of data.

4.7 Beacons (HELLO) and Links

HELLO messages are sent periodically between neighboring nodes to identify new links, measure packet loss between nodes and notice nodes that are not connected anymore. Links are formed between two network interfaces of two nodes, and two nodes connected to each other using two different media (Ethernet cable and Wireless LAN for example) may have two links connecting them. Each link is associated to a neighboring node it leads to, but each of the neighbor nodes may have several links associated with them, if they are connected to each other through both wireless LAN and wired Ethernet for example.

HELLO message beacons are broadcasted by a node from each physical network interface. Broadcasting intervals can be customized/optimized for any given network type, but must be consistent between the nodes in a network, so that each node knows when to expect a beacon from a neighboring node.

Nodes keep track of the connection status between two network interfaces using HELLO beacons. If the threshold for lost HELLO beacons is exceeded, the node that hasn't received the marks that link as disconnected, and it is not used until new HELLO beacons are received.

Links have a link cost, which is used to make intelligent decisions on how to forward data in the network. A link with a smaller cost is better than a link with a higher cost. The routing algorithm utilizes link costs in making decisions for the best route through several hops in the network (see 4.8 Routing Control Data Units and Route Selection). Several mechanisms, such as Wireless LAN signal strength, latency, bandwidth and others may be used to affect the link cost. By default, the MeshDriver does not use any of these measurements to aid in the routing because they are best optimized depending on the application. The default value for link cost is 1000.

In addition to cost, a link can have several states. The link can be:

- Connected, which is the normal operational state
- Disconnected, when there has been too many HELLOs lost (no data is sent to that link)
- Incompatible, if the MMP protocol version is incompatible (no data is sent to that link)
- Unauthorized, if access for the node the link belongs to is blocked (no data is sent to that link)

Each link also has an authentication status, based on the authentication process state or outcome (Handshaking, Unauthenticated, Authenticating, and Authenticated).

4.8 Routing Control Data Units and Route Selection

Whenever the nodes in a mesh network need to send data to a destination that is not directly connected (a neighboring node), they need to figure out the best route (through several mesh nodes).

The process (called route discovery) begins by the node, that wishes to send data, initiating a router request. A route request CDU is sent to the neighbors, who can forward the request to their neighbors or respond if they know the route to the requested destination. The response is sent in a route reply CDU. The route reply CDU is confirmed to be received in a route reply acknowledgement CDU. Once the route is known, the node that requested the route can proceed with sending the data.

A node can receive several route replies, if there are several possible routes to the destination.

The nodes will keep of track of the routes that are established in the network so that they don't have to send the requests all the time. If there are some changes in the network (a node becomes disconnected from it's neighbor, for example), a route error CDU is sent to notify the nodes in question.

The mechanism is based on AODV, which is detailed in RFC 3561. The exact CDUs that are different from the ones specified in the RFC but the principle is the same.

4.9 Ethernet master (ETHMASTER) Functionality

Due to the restrictions in current Ethernet standard, when non-mesh nodes are connected to mesh-nodes, only one mesh-node can serve the non-mesh nodes in a given Ethernet LAN segment as a gateway to the mesh network. The node that is serving the non-mesh nodes is named Ethernet master node.

ETHMASTER messages are used to negotiate which one of nodes connected to same medium (such as a LAN) servers the non-mesh nodes. The Ethernet master node is selected by the nodes collectively whenever none exists (when the first node in a network boots up or the existing Ethernet master node has left the network).

Ethernet master node selection process takes a few seconds which may cause breakage in communication to non-mesh nodes. Additionally some network topology changes may require expiration of ARP cache and routing of some of the non-mesh devices, which may result in a connectivity loss for a few minutes at maximum time when the master node is changed. The delay can occur only when several mesh devices are connected to the same LAN and the primary node serving the LAN is disabled, and concerns only non-mesh devices.

The Ethernet master nodes make note of every non-mesh node the notice. When there is a router request sent for an address that the Ethernet master node knows to be non-mesh and which it has a connection to, it can respond with a portal update CDU to notify the route requester that the destination is not a mesh node, and that it can be reached through given Ethernet master node in particular.

A node receiving a portal update message understands that the destination is reachable through the node that sent the portal update message, and will start forwarding the data to that mesh destination.

Using the Ethernet segment master functionality, two physically different LAN segments can also be connected to each other, forming one large layer combining the 2 segments. Whenever one of the nodes on either LAN segments receives regular Ethernet traffic, it converts it to Routed Data Units (see below) which are sent with the RDU destination address of a mesh node that manages the Ethernet segment. Since the mesh node is identified in the RDU, the mesh nodes that are along the route know where to send the data next.

4.10 Routed Data Units (RDU)

Routed Data Units are transferred from node to node one hop at a time to ultimately reach a destination further away. Whenever a node receives a Routed Data Unit, it consults its routing table and links, and selects the next hop to forward the RDU to.

Routed Data Units may contain several types of information, the most important of which is the "end user frame" type. End user frame RDUs are generated by an Ethernet segment master node whenever it receives non-mesh Ethernet frames – from another (non-meshed) device, or the IP stack on the mesh-enabled device itself.

Other RDU types include Mesh Ping, Mesh Node name Query and link list query. They are utilized through the use of the MDCTL interface that is used to control the driver behavior.

5. MeshDriver Product Keys

MeshDriver uses product keys for license accounting, providing system with unique node address and some specific characteristics.

From the end-user's perspective, product keys define the license under which the MeshDriver may be used. They may also contain information that alters the technical behavior of MeshDriver.

Appendix 1. Integration of Meshcom MeshDriver OEM

MeshDriver OEM ("MeshDriver"), is the implementation of Meshcom's mesh networking technology for device vendors and ISVs. It is designed from ground-up to be easily ported to new operating systems and hardware platforms. Current reference implementations are Linux 2.4, 2.6 and Windows XP version. Most of the code used in the implementation is shared between the operating systems.

MeshDriver is essentially media-neutral and does not depend on the underlying network type it receives data from; however routing can be greatly enhanced (especially in wireless) by providing additional information from the link layer such as wireless signal strength, bandwidth available and latency.

Some of the different media types present in modern networks impose their own limitations and requirements to operation, which need to be considered when integrating MeshDriver to a given platform.

1.1 Wireless LAN Operating Modes

Typical Wireless LAN architecture consists of one or more access points and clients, or end-users that connect to these access points. The inherent problem for mesh networking without any changes to the underlying 802.11 protocol is that clients can not communicate directly with each other, if used in infrastructure mode. Mesh networking can however be performed between access points in infrastructure mode, if a Wireless Distribution System (WDS), a method of access points to communicate with themselves, is available. In this mode the meshing is limited to the infrastructure network and clients cannot extend the network.

Another alternative in WLAN networks is to use ad-hoc architecture, where all clients can communicate directly with each other, but no access points exist. Ad-hoc can be used with MeshDriver to create multi-hop networks where all of the devices can participate in mesh routing. Typical access points do not support ad-hoc, so connecting off-the-shelf access points to ad-hoc networks is often impossible.

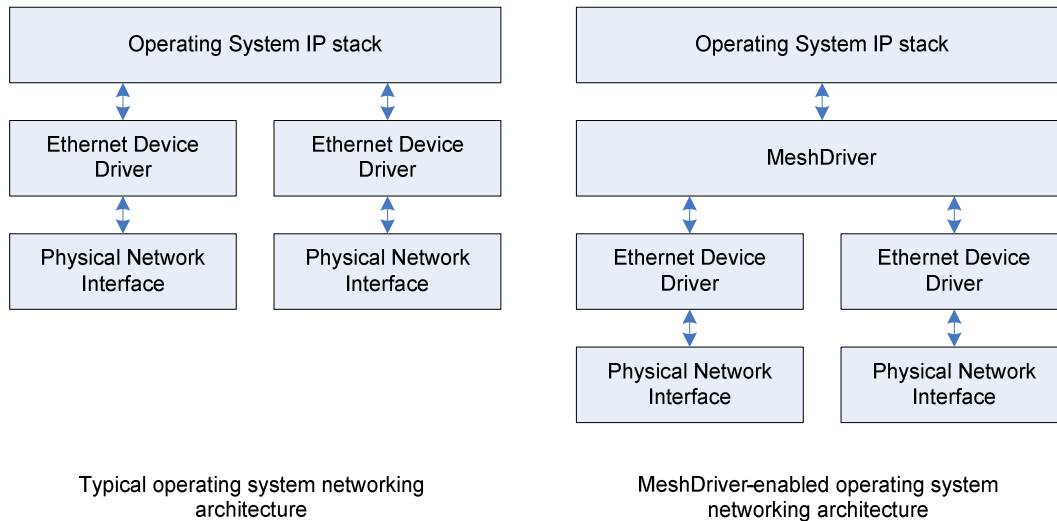
IEEE 802.11 standardization group has addressed these issues in task group s, which aims to enable multi-hop communication of both end-users and access points. The task group amendments to the standard will include changes to the radio-level functionality of the driver, so MeshDriver needs to be closely integrated with a hardware driver to achieve full 802.11s support.

1.2 Multi-hop Networks, Multi-Radio Systems

Wireless networks with several wireless hops and the possibility of several radio units in one device creates another challenge, since optimal network should use different radio channels or frequencies to eliminate interference between different hops. Since mesh networking enables many optional communication routes, the management of channels should also be dynamic. MeshDriver can be integrated with a radio-level driver to deliver information of which node the radio should be communicating with, but the exact algorithm for negotiating and selecting the channel to be used for each hop must be customized according to the characteristics of a radio network, if full support for meshing is required. Other alternatives is to use only one channel, which may cause problems with interference unless wired connections are also used, and using fixed channels, which limits the dynamicity of the network.

1.3 Software Integration

MeshDriver functions by taking control of the networking interfaces in a device. It receives Ethernet frames from a networking device driver, and forwards them either to another device driver, back to the same driver, or to the operating system's IP stack. The interface used to receive and send packets is operating system specific and needs to be implemented independently for different operating systems. The routing logic is platform-independent and can be compiled to different platforms with ease.



The operating system's IP stack is removed from the actual physical network interfaces and it only uses the virtual network interface provided by MeshDriver. Therefore any network topology changes in mesh are transparent to the IP stack and all existing IP-level protocols function regardless what actually happens underneath.

MeshDriver functionality is somewhat similar to bridging (802.1d) functionality present in some operating systems. The difference is that MeshDriver supports path selection based on advanced logic and measurement whereas 802.1d bridging assumes that it really doesn't matter by which route the frames get delivered.

1.4 Required Interfaces from Operating System

The fundamental interfaces required for MeshDriver operation is the possibility to send and receive Ethernet frames to devices and an IP stack, if one exists. MeshDriver also performs several operations at regular intervals, such as route advertisements and HELLO beacons, so timers are also required from the operating system. Some typical OS features are also expected, such as spinlocks, mutexes, and dynamic memory allocation.

Most applications also require the ability to configure MeshDriver in some way, or at least gather statistics. MeshDriver provides MDCTL interface for control and statistics. The exact way MDCTL interface is implemented is OS-dependant, and libraries for MDCTL are available for both Linux and Windows.

Appendix 2. Glossary and Acronyms

MMP	Meshcom Mesh Protocol, transfers Mesh control data and user data within the mesh network.
CDU	Control Data Unit, a container for different types of data exchanged in the mesh network. They are exchanged from a neighboring node to another.
RDU	Routed Data Unit, a unit of data which is transferred from non-neighboring node to another in a mesh network.
MDCTL	MeshDriver control interface.
AODV	Ad hoc on-demand distance vector, a routing algorithm utilized by MeshDriver