

MeshDriver - User Guide

Abstract

This document describes operation and configuration of MeshDriver for the end user. The versions covered in this document are Windows XP and Linux user-mode drivers.

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 introduction	3
2. Requirements.....	3
2.1 Windows Requirements.....	3
2.2 Linux Requirements.....	3
3. MeshDriver Security.....	3
3.1 Peer Authentication	3
3.2 Link-level Data Encryption.....	4
3.3 Limitations	4
4. Using MeshDriver	4
4.1 Example configuration	4
5. MeshDriver Installation	5
5.1 Installation on Windows XP	5
5.2 Installation on Linux.....	5
5.3 Running MeshDriver	5
6. mdcfg	Error! Bookmark not defined.
6.1 Driver version.....	Error! Bookmark not defined.
6.2 Ports	Error! Bookmark not defined.
6.3 Links	Error! Bookmark not defined.
6.4 Destinations	Error! Bookmark not defined.
6.5 Ping	Error! Bookmark not defined.
6.6 Node MAC and name	Error! Bookmark not defined.
6.7 Mesh ID.....	Error! Bookmark not defined.
6.8 Setpsk.....	Error! Bookmark not defined.
6.9 Authentication	Error! Bookmark not defined.
6.10 Access Control Lists.....	Error! Bookmark not defined.
7. Configuring Wi-Fi devices to Ad-hoc mode in Windows XP.....	6
8. Support and troubleshooting.....	14

1. MeshDriver introduction

MeshDriver provides self-organizing, self-healing networking functionality in an easy-to-install software package for end-users and system integrators. Meshcom MeshDriver utilizes devices that you already have, no new hardware is needed! With MeshDriver the users are able to:

- Create community networks for sharing internet connections and local information without the need to invest in infrastructure.
- Expand the network and take part in communications by just installing the software.
- Create/Expand/Replace network infrastructures without the need for configuration at a very low cost and a fast roll-out speed.
- Build new applications for dynamic and mobile environments!
- Increase security of Ad-Hoc wireless networks.

Note: Being a software-only solution MeshDriver relies on established connectivity between devices. For example, it does not automatically configure wireless adapters and establish wireless connections. Using wireless adapters in Ad-Hoc mode is one way to establish wireless connections automatically.

2. Requirements

2.1 Windows Requirements

- Windows XP operating system, preferably with Service Pack 2
- At least one network adapter

2.2 Linux Requirements

- Linux kernel 2.6.x
- Required kernel features:
 - CONFIG_PACKET
 - CONFIG_UNIX
 - CONFIG_TUN
- Glibc version 2.3.5 or newer.

The Mesh device (seen as mesh0 by the user) is implemented with the Universal TUN/TAP driver as a TAP-device. It allows packet reception and transmission on OSI layer 2 for user-space applications. For packet reception and transmission from/to other Ethernet interfaces MeshDriver uses the raw packet socket AF_PACKET.

Note! MeshDriver might work with other than above indicated versions of kernel and Glibc, but they are currently the only tested ones.

HINT!

There is a separate kernel-mode driver existing for Linux, which has better performance figures and thus is better suited for devices with limited CPU resources.

3. MeshDriver Security

To provide communications integrity and confidentiality MeshDriver supports pre-shared key peer authentication and link-level data encryption. It can be enabled in insecure networks, like wireless Ad-Hoc networks, for which there's no security standard.

3.1 Peer Authentication

There are two authentication modes supported by MeshDriver:

Open. In this mode an identity of the peer is not checked and the link becomes automatically authenticated.

SRP. In this mode Secure Remote Password protocol (SRP, RFC 2945) is used to verify authorization of the parties by proving the mutual knowledge of a secret phrase or key (AKA

pre-shared key authentication). In addition a Pairwise Master Key (PMK) is derived for link-level encryption.

Each network interface controlled by MeshDriver can be configured to use either Open or SRP authentication mode. Default authentication mode is Open.

HINT!

There is no need to enable MeshDriver authentication on the interfaces that use some other security solutions, like WPA/WPA2.

3.2 Link-level Data Encryption

If a network interface is configured to use SRP authentication, then all links established through this interface will be encrypted with 128bit AES algorithm in CCM mode following IEEE 802.11i standard. Encryption key is derived during the authentication phase and is unique for each link.

NOTE:

Authentication without encryption is not supported since only encryption can guarantee that the arrived packet really comes from authenticated node.

Encryption without authentication is not supported since it is the authentication that allows to derive cryptographically secure encryption key. Besides there's no sense to omit the authentication step.

3.3 Limitations

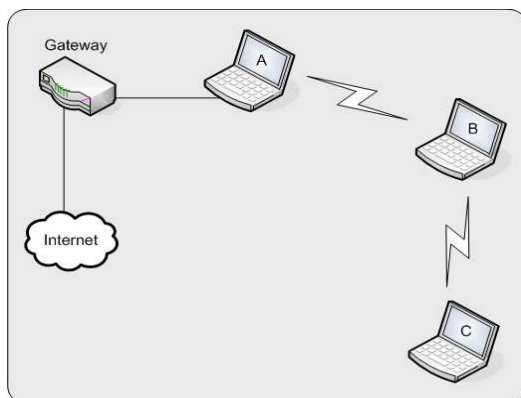
There are a number of limitations to the current MeshDriver security framework:

- Broadcast messages are not encrypted.
- Link encryption key is derived only once per link lifetime.
- There is no peer-to-peer encryption (for peers which are more than 1 hop away).
- Encryption consumes a lot of computational resources and may lower the throughput of the node.

4. Using MeshDriver

4.1 Example configuration

MeshDriver can be used to connect laptops equipped with a wireless adapter to a wired network. When at laptop A with MeshDriver running is connected to the Internet, it connects Laptop B with MeshDriver to the Internet as well.



When laptops A, B and C with MeshDriver are present, they can use each other to make a communication link to the Internet. Unlike traditional WiFi networks (consisting of clients communicating with centralized access points), all devices extend the reach of the network by relaying traffic from device to another. In the example here, laptop C can communicate with all of the devices in the network through laptop B. The network is formed automatically and adding new devices does not need any configuration.

5. MeshDriver Installation

5.1 Installation on Windows XP

Installation is simply done by running the installation package. Installer will check for previous versions of MeshDriver installed, and note you to remove them before installing the newer version. Installation process takes about 1-2 minutes. After the installation is done, the installer asks you to reboot the computer to finish the installation process.

Uninstallation can be done by clicking *Start Menu -> Meshcom MeshDriver -> Uninstall MeshDriver* or opening *Control Panel -> Add or Remove Programs* and clicking remove after selecting Meshcom MeshDriver from the list.

5.2 Installation on Linux

To run the installation on Linux you must set execution rights on the installation script. After that you can execute the installation as root.

```
# chmod +x meshdriver_1.0_linux_ia32.sh
# ./meshdriver_1.0_linux_ia32.sh
```

The script will ask you to accept the end-user license agreement and prompt for installation directory (the default is `/opt/MeshDriver/`).

5.3 Running MeshDriver

By default MeshDriver for Linux will bind to all available Ethernet interfaces that are up. To restrict binding to specific interfaces supply MeshDriver with the interface names. You can specify run-time options for meshdriver by adding them `/etc/meshdriver/meshd_options`.

Example: `/etc/meshdriver/meshd_options` file that restricts binding only to `eth0` and `eth1`:

```
--config /etc/meshdriver/meshd.conf eth0 eth1
```

After making changes to the options file the driver has to be restarted. This can be done by running:

```
# /etc/init.d/meshdriver restart
```

After MeshDriver is started the IP-addresses from binded interface should be removed and an IP address should only be assigned to the mesh device, `mesh0`.

For example if `eth0` is configured as `192.168.1.1`:

```
# ifconfig eth0 0.0.0.0
# ifconfig mesh0 192.168.1.1
```

The first command removes ip-address from `eth0`, and the second one assigns the ip address previously to `mesh0`.

Linux driver can be stopped with the following command:

```
# /etc/init.d/meshdriver stop
```

MeshDriver will log important events to syslog (usually /var/log/daemon.log, check your syslog configuration). In problem situations use the mdcfg utility and syslog for troubleshooting.

6. mdcfg

MeshDriver comes with a command line tool 'mdcfg' which can be used to monitor available ports, network links and routes. It also has functionality to do native mesh ping. When called without parameters, mdcfg tool prints out available commands and parameters.

NOTE: Using mdcfg on Linux requires superuser privileges.

Example:

```
# mdcfg
Usage: ./mdcfg [options] <command> [arguments]

Options:
  -v          Output tables of links and destinations in full format.
  -t          Output tables of links and destinations in compact
              format (default).
  -m          Output in machine readable format.

Commands:
  help          Display this help and exit.
  version       Display driver version.
  ports         Display list of ports.
  links [port] Display list of links under all ports or under the
                specified port.
  destinations  Display all destinations seen on this node.
  ping <mac>    Ping a node.
  traceroute <mac> Traceroute a node.
  nodemac       Display this node's MAC address.
  getnodename   Display this node's name.
  setnodename <nodename>
                Set this node's name.
  getmeshid     Display mesh network id.
  setmeshid <meshid> Set mesh network id.
  meshlist      List available mesh networks
  getdefaultauth Get default authentication type.
  setdefaultauth <type> Set default authentication type. Run without arguments
                        for list of types.
  setpsk [<key>] Set preshared key.
  setportauth <port> <auth type mesh> <auth type virtual>
                Set port authentication type. Run without arguments for
                list of types.
  setportflushtime <port> <timeout>
                Set the interval for flushing frame buffers of mesh
                links for the specified port.
  setacl <white|black> Set local access list type.
  addacl <addr>       Add entry to local access list.
  delacl <addr>       Delete entry from local access list.
  showacl           Show current access list configuration.
  getlinktimeout    Gets the time of inactivity until a link becomes
                    disconnected (ms).
  setlinktimeout <time> Sets the time of inactivity until a link becomes
                    disconnected (ms).
  setlinkcost <portname> <linkmac> <cost>
                Set the cost for the mesh link with <linkmac>,
                found under the specified port
  admkeygen [file name] Generate public/private Administrator key pair.
  getlicense        Show current MeshDriver license.
  aodvcostdelay [true|false]
                Shows if cost delay for AODV routing is enabled or sets
                it if the argumen is given.
  aodvrefreshinterval [interval]
```

```
Shows the current AODV route refresh interval (ms) or
sets it if given as argument. 0 is off, minimum is 5s.
```

The above list shows all available parameters a user can use.

6.1 Driver version

Displays the source version of the running driver. Helpful for bug reports and support requests.

Example:

```
# mdcfg version
MeshDriver 1.0.7
```

6.2 Ports

Ports command shows currently active ports eg. Ethernet devices that are bound to mesh network. This includes physical adapters and virtual mesh device itself.

Example:

```
# mdcfg ports
Id Name   ESM    B/w M.Auth V.Auth Media state Description
-----
  1 eth1   Yes   21600 Default Default Connected Adapter
  2 vlan0 Yes    n/a Default Default Connected Adapter
# mdcfg -v ports
Port [1]:
  Name:                eth1
  Description:         Adapter
  Ethernet segment master: Yes
  Bandwidth:           21600
  Mesh authentication: Default
  Virtual authentication: Default
  Media state:         Connected
Port [2]:
  Name:                vlan0
  Description:         Adapter
  Ethernet segment master: Yes
  Bandwidth:           n/a
  Mesh authentication: Default
  Virtual authentication: Default
  Media state:         Connected
```

Each port description consists of:

- Index number. This Index number should be used when passing port value to links command. See 'links' command below.
- Port name.
- Ethernet Master mode. Ethernet Master node relays network packets to and from virtual nodes. There can be only one Ethernet Master per Ethernet segment.
- Port bandwidth. Either reported by the driver or estimated, like 21600 Kbps for a 802.11g interface.
- Port authentication methods. See chapter on MeshDriver security.
- Media state (connected/disconnected). Note that some NICs are reported always connected by the OS, even if a cable is unplugged
- Description of the device

6.3 Links

Links command prints out all the links to neighbor nodes from the given port. If active node is Ethernet Master, the list will include a list of virtual nodes, not just nodes that have mesh network enabled. If the node is not Ethernet Master, only mesh nodes are shown in the link list.

Example:

```
# mdcfg links
Index Node address      Interface MAC    State Auth  Age    SQF Err    B/w    Cost
-----
      1                7A:2A:DF:81:F3:64 INC HS     50    1.00  0     n/a    n/a
      3                96:3B:85:DE:4F:C4 CON OK    85960  1.00  0     n/a    1000
# mdcfg -v links
Link [1]:
Node MAC:      00:00:00:00:00:00
Link MAC:      7A:2A:DF:81:F3:64
State:         Incompatible node
Auth. State:   Handshaking
Age:           180ms
Signal Strength: 1.000
Errors:        0
Bandwidth:     n/a
Cost:          n/a
Link [3]:
Node MAC:      00:00:00:00:00:00
Link MAC:      96:3B:85:DE:4F:C4
State:         Connected
Auth. State:   Authenticated
Age:           88090ms
Signal Strength: 1.000
Errors:        0
Bandwidth:     n/a
Cost:          1000
```

Link description consists of:

- Index number
- Node MAC. A MAC-address that uniquely identifies the node in the mesh network.
- Interface MAC. A MAC-address of the node's network adapter through which the link to the node was established.
- Link state. Link connectivity state can be:
 - Connected: A network packet from the neighbor node has been recently received
 - Disconnected: The neighbor node stopped responding to HELLO beacons.
 - Incompatible: The neighbor is using incompatible communication protocol.
 - Unauthorized: The neighbor node is not allowed by ACL.
- Authentication status. Can be
 - Handshaking: initial handshake to agree on authentication method.
 - Authenticating: authentication in progress.
 - Authenticated/Unauthenticated: result of authentication.
- Age. How long ago a packet was seen from this neighbor.
- Signal strength (when supported by wireless adapter driver).
- Number of transmission errors.
- Bandwidth of the link (when supported by wireless adapter driver).
- Link cost.

6.4 Destinations

Destinations command will print out each node that is known to be accessible at this time. Only nodes that were recently accessed are listed. If the node is not in the list it does not mean that it is not accessible. It may have expired from the destinations table.

Example:

```
#mdcfg destinations
Index Destination      Age(ms)  Link Hops  Cost
```

```
-----  
0 96:3B:85:DE:4F:C4 1470 => 22:7B:83:E0:FF:FE  
  
# mdcfg -v destinations  
Destination [0]:  
  Dest. MAC: 96:3B:85:DE:4F:C4  
  Proxy MAC: 22:7B:83:E0:FF:FE  
  Age:      64590ms
```

In the example above destination is a virtual node (not running MeshDriver) that is accessible through nodes 22:7B:83:E0:FF:FE.

6.5 Ping

A ping can be used to show connectivity between two nodes. Pass the remote node's MAC as parameter. Note that non mesh devices do not reply to ping requests.

Example:

```
# mdcfg ping 3A:C7:2D:74:32:02  
Pinging node 3A:C7:2D:74:32:02 ... Press CTRL+C when done.  
Reply from 3A:C7:2D:74:32:02: id=0 time=1ms  
Reply from 3A:C7:2D:74:32:02: id=1 time=1ms  
Reply from 3A:C7:2D:74:32:02: id=2 time=1ms  
Reply from 3A:C7:2D:74:32:02: id=3 time=1ms  
Reply from 3A:C7:2D:74:32:02: id=4 time=1ms
```

6.6 Traceroute

traceroute command is similar to usual IP traceroute. It traces the path of the packet to a target node and prints out a list of nodes passed by the packet. Only mesh nodes can be a target to traceroute at this moment.

Example:

```
# mdcfg traceroute 00:FF:52:FC:BE:DF  
Traceroute to node 00:FF:52:FC:BE:DF...  
1: 00:FF:52:FC:BE:DF 37.158ms
```

6.7 Node MAC and name

Each mesh node is identified by a unique MAC address. It either comes with a license key or is randomly generated. In addition each node can be assigned a symbolic name which is easy to remember.

Example:

```
# mdcfg nodemac  
Node MAC: FA:36:01:E9:A3:F2  
  
# mdcfg getnodename  
Nodename: mynode  
  
# mdcfg setnodename livingroom  
Nodename set to 'livingroom'.
```

6.8 Mesh ID

Mesh ID serves the same purpose as ESSID in WiFi – it allows to differentiate between spatially overlapping mesh networks (e.g. a group of wirelessly connected laptops) and establish links only with

nodes from the same mesh network. `mdcfg` allows to set and to check mesh ID and also to see what mesh networks neighbor nodes belong to.

Example:

```
# mdcfg getmeshid
MeshID is 'MYMESH'.
# mdcfg setmeshid foo
MeshID set to 'foo'.
# mdcfg getmeshid
MeshID is 'foo'.
# mdcfg meshlist
Node is part of mesh network 'l07'.
Node sees following outside mesh networks:
  'foo' (open)
  'MYMESH' (open)
```

6.9 Setpsk

'`setpsk`' command allows to set pre-shared key/pass phrase for SRP authentication. It can take the key as an optional argument or ask it interactively. In the latter case the key is not displayed while it is typed in.

Example:

```
# mdcfg setpsk mysecret
Preshared key is set
```

6.10 Authentication

'`setportauth`' command allows to set authentication modes for particular port. One has to set authentication modes for mesh and virtual (non-mesh) nodes separately, hence two required fields. Setting non-open for virtual nodes prevents the node from establishing links to virtual nodes.

'`setdefaultauth`' can be used to set default authentication policy.

Examples:

```
# mdcfg setportauth 4 SRP SRP
# mdcfg setdefaultauth Closed
```

6.11 Access Control Lists

Another way to control which nodes can establish links with that node is to create Access Control List (ACL). ACL can be a white list, or a black list. In the case of a white list - only nodes whose MACs are present in the list will establish links with. In the case of a black list the nodes whose MACs are present in the list will reject links with. The following commands allow you to manipulate the ACLs.

Example:

```
# mdcfg setacl black
ACL type set to blacklist.
# mdcfg addacl 01:02:03:04:05:06
ACL Updated.
# mdcfg showacl
ACL type: Blacklist
ACL entries:
  <01:02:03:04:05:06>
# mdcfg delacl 01:02:03:04:05:06
ACL Updated.
```

```
# mdcfg showacl
ACL type: Blacklist
ACL entries:
No entries in access list.
```

6.12 Link Timeout

Link timeout can be adjusted to enhance the mesh network performance. A minimum value of 600 can be used for MeshDriver 1.0.2 and later networks, but for networks older MeshDrivers the default value of 2000 is needed.

Example:

```
# mdcfg setlinktimeout 600
Link idle timeout set to 600 ms.
```

6.13 Routing based on Link Cost

To give greater flexibility and more options to the users, MeshDriver supports cost-based routing. The driver's routing protocol (AODV) already selects the fastest route but this feature makes sure that bad links will continuously be avoided. Cost-based routing can also be used to force choosing a certain routing path among several others that are similar or even slower. MeshDriver does not calculate the link cost, a value of 1000 is used by default. Different values for link costs can be set from outside the driver using a mdcfg command.

Enabling and disabling.

Link costs take effect only if the "aodvcostdelay" feature is enabled. Otherwise there is no impact on routing.

To enable routing based on link cost:

```
# mdcfg aodvcostdelay true
```

Use 'false' instead of 'true' to disable it.

To check current status:

```
# mdcfg aodvcostdelay
AODV cost delay is enabled.
```

Setting the link cost.

To set the link cost:

```
# mdcfg setlinkcost eth0 00:18:39:BF:AB:FE 10000
```

The <linkmac> argument is the interface MAC, not the node address. The new cost can be checked with the "mdcfg links" command. Cost changes will not affect routes immediately, for details see the "aodvrefreshinterval" command.

Route refresh interval.

The route refresh interval determines how often valid/used routes are updated. Therefore it directly affects the cost-based routing in case of ongoing traffic: new costs on the network will not take effect until the route is refreshed. The lower the value, the more often the routes are refreshed and also there will be more routing traffic.

The default value is 15s and minimum is 5s.

To check the route refresh interval:

```
# mdcfg aodvrefreshinterval
AODV route refresh interval is 15000 ms.
```

To set a new route refresh interval:

```
# mdcfg aodvrefreshinterval 10000
```

NOTES:

- If you want a specific connection (link) not to be used for routing anymore you should set higher costs for this link on both nodes.

- If you want a specific node not to be used for routing anymore you need to set the cost higher on some or all links of that node.

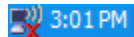
7. Configuring Wi-Fi devices to Ad-hoc mode in Windows XP

Since MeshDriver relies on established connectivity between computers it may be desirable to configure Wireless LAN card on a Windows XP laptop to Ad-Hoc (computer-to-computer) mode to fully utilize mesh networking. The benefits of Ad-Hoc mode are:

- Wireless connections to neighbor laptops are established automatically.
- A single WLAN card can have several connections at once.

By default, the mode is not enabled and the following steps need to be performed:

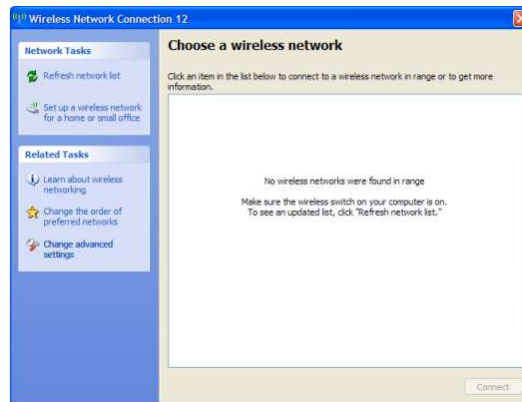
Step 1. After you insert the WLAN card (or boot when the card is already inserted), you should see a wireless LAN icon in the task bar.



NOTE: The icon can also be present without the red cross.

Step 2. Right-click on the icon and select "View Available Wireless Networks".

You should be presented with the following screen.



NOTE! The pictures in this chapter present the dialogs that English Windows XP with Service Pack 2 uses to configure wireless connectivity. They may look different on previous versions.

The screen can also list other networks in the wireless range. If this is the first laptop you are configuring for mesh networking, you must set up the network first. If not, you should see an ad-hoc network displayed and can just select it by clicking it.

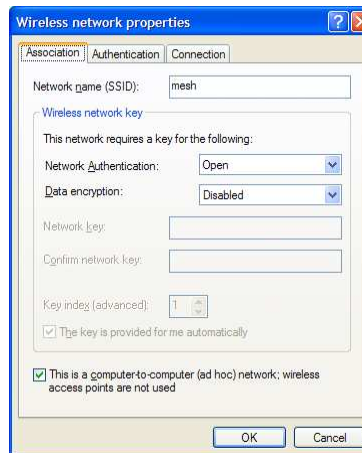
Step 3. If you haven't set up ad hoc network, you need to click on "Change the order of preferred networks" text.



After that, you should see wireless network connection properties screen.

In the screen, there may be some networks listed under "Preferred networks:", if you have used one or more previously.

Step 4. Adding of an ad-hoc network is performed through the 'Add' button in Preferred Networks section. The settings need to be configured as follows:



Network name (SSID):
Select a name for your mesh network ("mesh" in our example)

Network Authentication:
Open

Data Encryption:
Disabled

This is a computer-to-computer (ad hoc) network:
checked

After you have entered the information, press "OK".

Step 5. You should be displayed with the previous page again, with a new network in the preferred networks list.

After these settings, you should be done. Once enabled in one laptop, other laptops should find the network in the "View Available Wireless Networks" and it can be selected from the list.



8. Support and troubleshooting

In problem situations, first visit EmbedOne community website (<http://embedone.com>) for updated documentation and source code. If your question isn't answered there, don't hesitate to post questions to our support forums!