 Agenda

• A closer look at the RFC

• Implementation overview
 RFC3626 divides the functioning of OLSR into two groups

**core functioning** - always required for the protocol to operate

**auxiliary functioning** - provides additional functionality, which may be applicable in specific scenarios.
Core repositories

OLSR core functioning maintains the following repositories:

- Multiple Interface Association set
- Link set
- Neighbor set
- 2hop neighbor set
- MPR set
- MPR selector set
- Topology information base
- Duplicate set
Note that a OLSR *packet* can contain multiple OLSR *messages*.
Nodes maintain their link, neighbor, 2-hop neighbor and mpr sets based on HELLOs
Core messages - TC

Nodes advertise link state using TC messages.

<table>
<thead>
<tr>
<th>Bits:</th>
<th>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ANSN</td>
</tr>
<tr>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td>Advertised Neighbor Main Address</td>
</tr>
<tr>
<td></td>
<td>Advertised Neighbor Main Address</td>
</tr>
</tbody>
</table>

Nodes maintain their topology set based on TC messages.
Core messages - MID

If a node runs OLSR on multiple interfaces it generates MID messages.

Nodes maintains a Multiple Interface Association set based on received MID messages.
Non-OLSR interfaces - HNA messages

External connectivity is offered by sending HNA messages.

A node maintains a HNA set based upon this information.
Link quality

Hysteresis

Link hysteresis is a more robust link-sensing scheme. It is used in addition to the core link-sensing.

Hysteresis ensures a certain stability for links to be considered symmetric. Links are also detected as asymmetric quicker using the suggested algorithm and values.

Link layer information

The RFC suggests that one should utilize information from the link-layer if possible.

Very loose terms used. Could be used in link sensing and MPR calculation.
Tweaking parameters

Redundant topology information

- To offer a more robust topology understanding nodes can include more than just its MPR selectors in TC messages.
- Include all selected MPRs or include all neighbors.
- Results in more bandwidth usage.

MPR redundancy

- For more robustness in a network nodes can be set to have a higher coverage parameter when calculating MPRs.
- By default all 2 hop neighbors are to be covered by minimum one MPR. This parameter can however be increased resulting in (in most cases) more neighbors being chosen as MPRs.
- But this results in a less optimized flooding scheme!
Implementing OLSR

http://www.olsr.org

• History
• Challenges
• Basic structure
• Plug-ins
• Installing and running
• Real life usage
Implementing OLSR - Brief history

Spring -03:
  Started work on master thesis at UniK. Modified a INRIA implementation to add MID functionality and fix HNA functionality. Also started working on a GUI.

Autumn -03:
  Added Ipv6 support and started moving towards RFC compliance. This included rewriting all code(in the end). First public release 0.2.0

Spring -04:
  www.olsr.org registered and set up. 0.4.0 released - full RFC compliance. Plugin interface implemented. Various plugins implemented.

Summer -04:
  Master thesis at UniK finished. olsr.org became a community driven open source project. Thomas Lopatic joined as developer. Windows and FreeBSD port and various updates. (0.4.7)

Autumn -04:
  OSX port and NetBSD port. ETX like Link Quality routing implemented. Much more flexible configuration options.(0.4.8)

Spring -05:
  Coming soon 0.4.9: New interesting plugins, bugfixes, optimizations…
Technical

- Implementation for GNU/Linux, FreeBSD, NetBSD, OSX and Windows systems
- Implemented in pure C
- All areas of the RFC covered
- Supports IPv4 and IPv6 (Linux)
- Plugin support (dynamically linked libraries)
- Supports dynamic updates of NICs
- Very flexible configuration options
- Includes platform independent HTTP front-end
- Licensed under a BSD-style license
Hardware on which olsrd runs

Standard i386 and PPC based computer systems.

The MIPS based WLAN accesspoint WRT54G from LinkSys(Cisco) and the MeshCube

The strong-ARM based iPAQ from Compaq/HP

PS. It runs on NetBSD ;-) “Of course it runs NetBSD”
Statistics

• www.olsr.org has been visited by over 60000 daily unique guests since January -04.

• Over 9000 downloads of the code has been done during the same period.
Some challenges - basic OLSR

- Directing output to specific interfaces, binding sockets to devices (255.255.255.255 broadcasts)
- Most possible transparent code considering IPv4/IPv6
- Fast indexed data storage structures
- Most possible modular design of all components
- As much run-time and dynamic initialization as possible
- Implementing support for loadable plugins (DLLs) and designing a interface for DLL communication
- Optimized code (embedded systems)
- OS dependent code separation
Basic overview

Main design principal: *modularity*

As much run-time initialization as possible!
Loadable plugins

- No need to change any code in the olsr daemon to add custom packages.

- Users are free license plugins under whatever terms they like.

- Plug-ins can be written in any language that can be compiled as a dynamic library. GNU/Linux even allow scripts!

- No need for people with extended OLSR versions to rely on heavy patching to maintain functionality when new olsrd versions are released.
Dynamically loadable plugins

A plugin (dynamically loaded library) is a piece of code that can be linked and "loaded" by an application in runtime.
Link Quality extention

OLSR only uses hop-count as metric. Even if using hysteresis unstable links will often be chosen for routes.

Thomas and the c-base crew has developed and implemented and experimental ETX(\textit{expected transmission count})-like link quality routing.

Uni-directional link-quality metrics based on packet loss, is distributed throughout the network. Routes are then calculated based on these metrics using Dijkstra.

Shows much more stable routes in real-life networks!

Read more at olsr.org
Getting and building olsrd

You can download the latest stable versions from olsr.org both as source and binary versions. Installers/packages are available for many systems.

If you want bleeding edge use CVS:

$ cvs -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/olsrd login
$ cvs -z3 -d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/olsrd co olsrd

To build you need all the usual tools(make, makedep gcc, glibc etc.)
For win32 you need cygwin

$ make OS=linux
$ make install

Now to configure edit the config file:

$ vi /etc/olsrd.conf
Running olsrd

Various debug levels from 0(background proc) to 9
If ran with debug level 1 it generates human readable output
HTTP-info plugin

Implements a small HTTP server that delivers dynamic content to clients. An experimental administration interface is also available where one can updates settings in real-time.
Real-life applications

Olsrd has gone from a research-based implementation to a real-life routing daemon!

The *c-base community* in Berlin is much involved in the development. They run a large real-life free community network.

Many other community(and scientific) networks use olsrd as their main routing daemon.
c-base network snapshot

Created using dot-draw plugin
Future work

Next release (0.4.10):
- IPv4/6 simultaneously
- IPv6 on *BSD
- More run time configuration options

Hard to say much on long term goals as this project is basically about having fun and learning. New features are implemented on request and by need. Focused on real-life usage!

For others:
- Security
- Auto-configuration
- Internet-connectivity
Questions?