



Optimized LinkState Routing

Andreas Tønnesen
andreto@olsr.org

www.olsr.org



Agenda

- A closer look at the RFC
- Implementation overview



A closer look at RFC3626

- Features
- Core* functions
- Auxiliary* functions



OLSR - Optimized LinkState Routing

A table-driven, pro-active protocol

Optimized by MultiPointRelay flooding and messaging

Generates a constant overhead of control traffic

No route lookup delay



RFC3626 - OLSR

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.



Information repositories

In a table-driven protocol such as OLSR, basically *everything* is related to tables(repositories, databases).

These tables need to be maintained both upon receiving information and regarding the time received information is to be considered valid(timeout).

All route calculation and most packet generation is based on these tables.



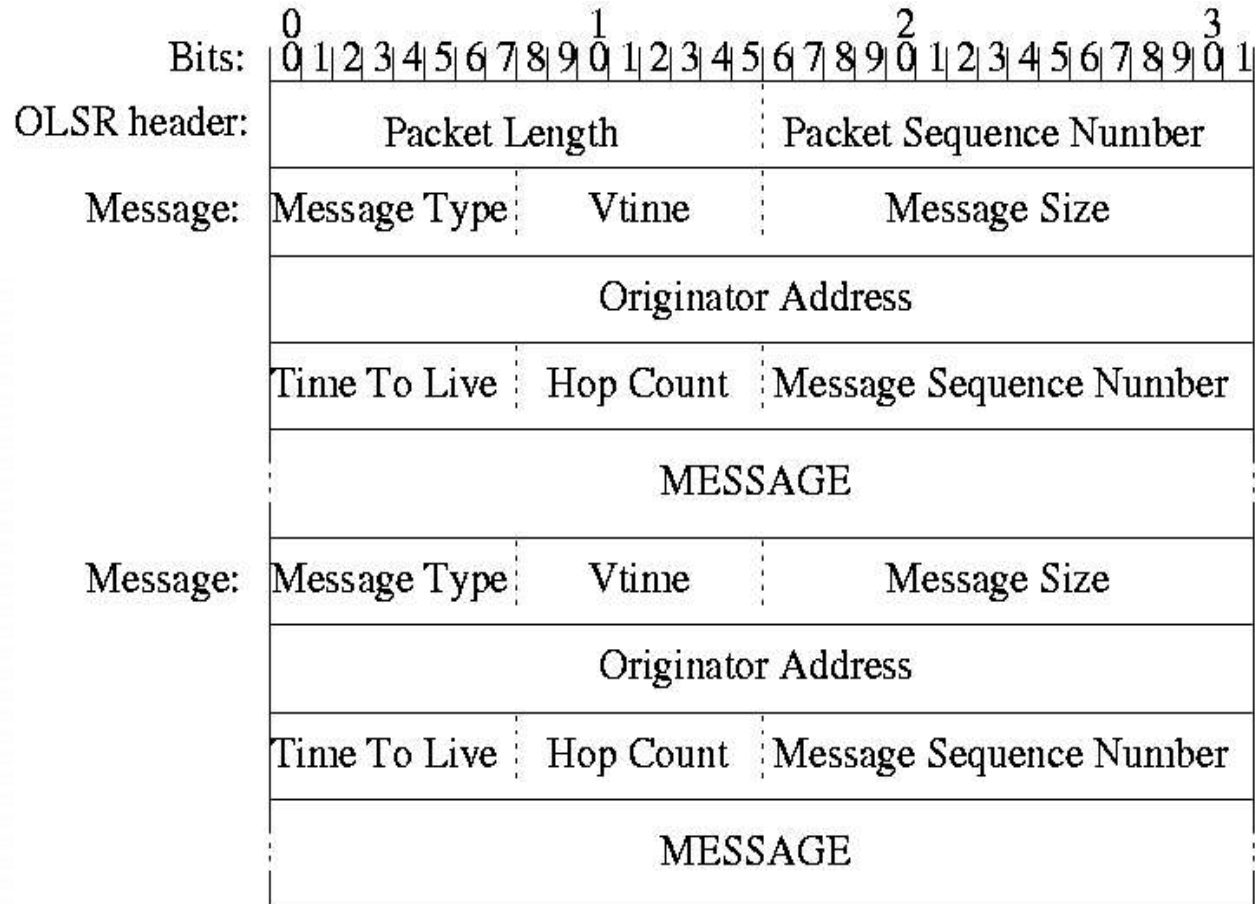
Core repositories

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



OLSR packet format





Forwarding messages

OLSR defines a *default forwarding algorithm* ensuring that all known **and unknown** message-types are forwarded according to the MPR optimization.

To avoid synchronization when forwarding a jitter is used. This means that a message is to be cached in the node for a random time interval before forwarding it.

Due to this messages are often “piggybacked” - one OLSR packet contains several OLSR messages.



Core messages - MID

In OLSR a nodes IP address is used as a identifier. So all nodes must set one *main address*.

If only using one interface this interfaces IP address (IP header source) is used.

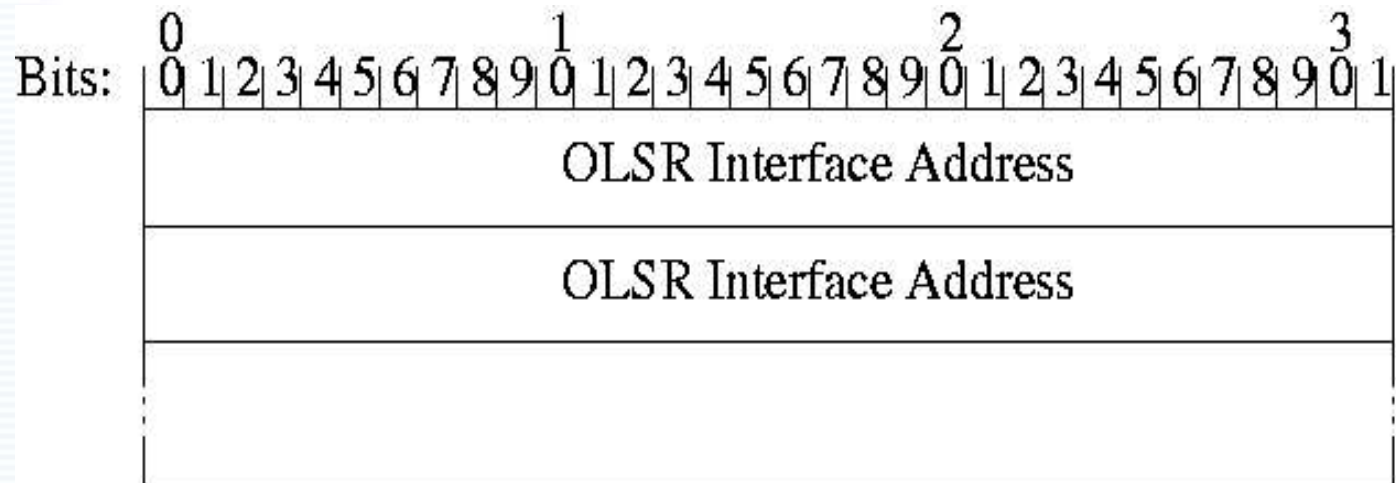
If running OLSR on multiple interfaces *one* address must be chosen. And used as originator in all OLSR packets.

To announce the usage of several interfaces(IPs) *Multiple Interface Declaration* messages are flooded.



Core messages - MID

MID messages are basically just a list of addresses



Nodes maintains a Multiple Interface Association set based on received MID messages.



Core messages - HELLO

Neighbor-sensing is an important part of a MANET routing protocol.

OLSR does neighbor-sensing by emitting regular HELLO messages.

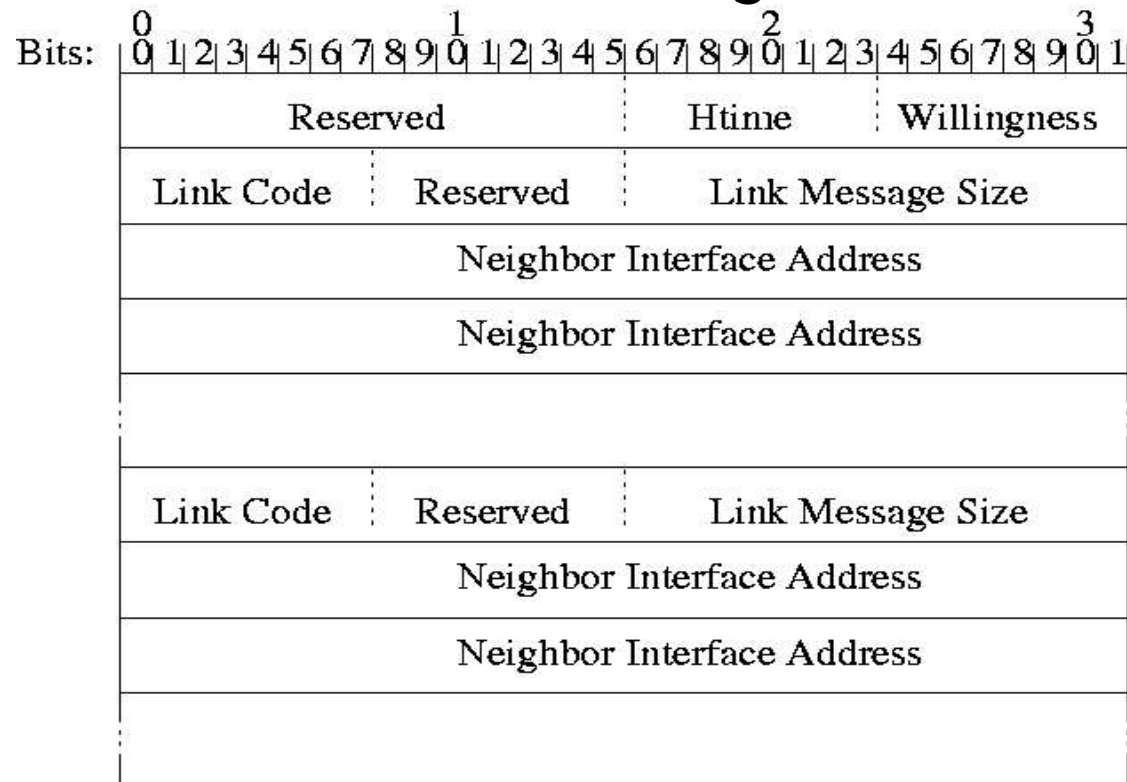
However - HELLO messages have multiple functions:

- Link sensing
- 2 hop neighbor sensing
- MID selector registration



Core messages - HELLO

Neighbors with same status are grouped to preserve bit usage. HELLO messages are generated on a per interface basis due to link-sensing.





Core messages - TC

A node announces its link-set by flooding *Topology Control* messages.

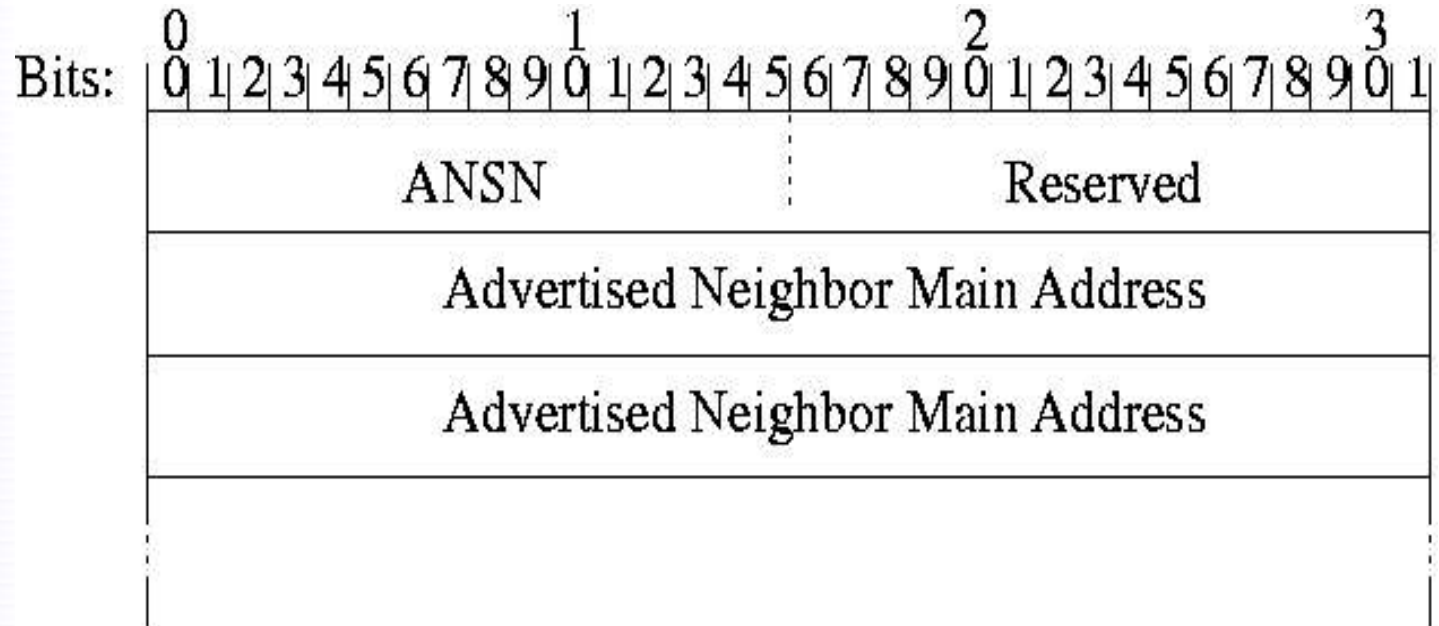
As an optimization a node need only announce its MPR-selectors. This is the “second MPR optimization”.

TC messages uses an *Advertised Neighbor Sequence Number* to announce the “freshness” of the announced link-set. It is incremented by 1 if the announced link-set is changed from the last transmitted TC message.

Sent on a regular interval - but also triggered by link-set changes!



Core messages - TC



Nodes maintain their topology set based on TC messages.



MPR calculation

MPRs are calculated so that a node can reach all its symmetric 2 hop neighbors via one of its MPRs.

MPR calculation is based on willingness announced by neighbors.

MPR calculation is based on the information from:

- The neighbor set
- The link set
- The 2 hop neighbor set



Route calculation

Routes are calculated using a rather trivial algorithm
But we will not get into it here :-)

Route calculation is based on the information from:

- The MID set
- The link set
- The neighbor set
- The 2 hop neighbor set
- The TC set



Auxiliary functioning

The auxiliary functioning is based on five main sections:

- Non-OLSR interfaces(external access)
- Link-layer notification
- Link-hysteresis(robust link-sensing)
- Redundant topology information
- MPR redundancy



Non-OLSR interfaces - HNA messages

Nodes in the OLSR network might have access to other networks as well. Typically a node could have Internet access through a ethernet link(not running OLSR).

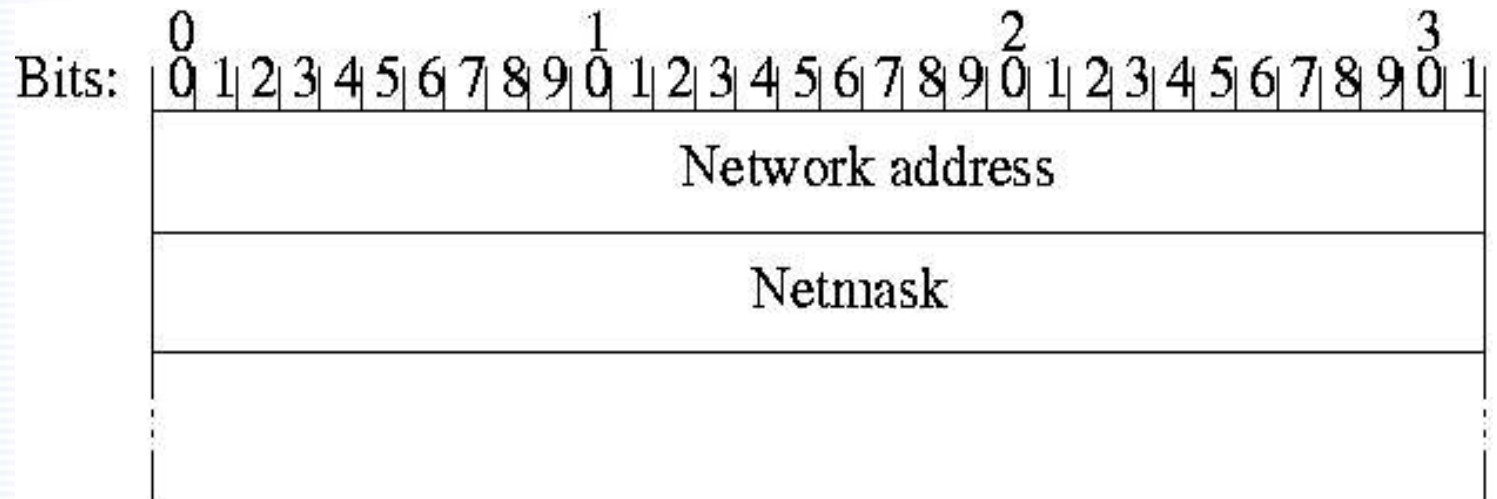
A node can announce itself as a gateway from the OLSR network to other networks using *Host and Network Association* messages.

These HNA messages need not be supported by all nodes in the OLSR network - they are still forwarded due to the default forwarding algorithm.



Non-OLSR interfaces - HNA messages

HNA messages are just a list of network addresses and netmasks.



A node maintains a HNA set based upon this information



HNA route calculation

HNA routes are added to the routing table based on hop-count and are dynamically updated as the topology changes.

That is, if multiple routes to the same network exist than the route with the smallest hop-count is used.

These routes must be added using the nexthop as the gateway - this leads to the situation that a node in effect cannot choose what gateway to use.



Link hysteresis

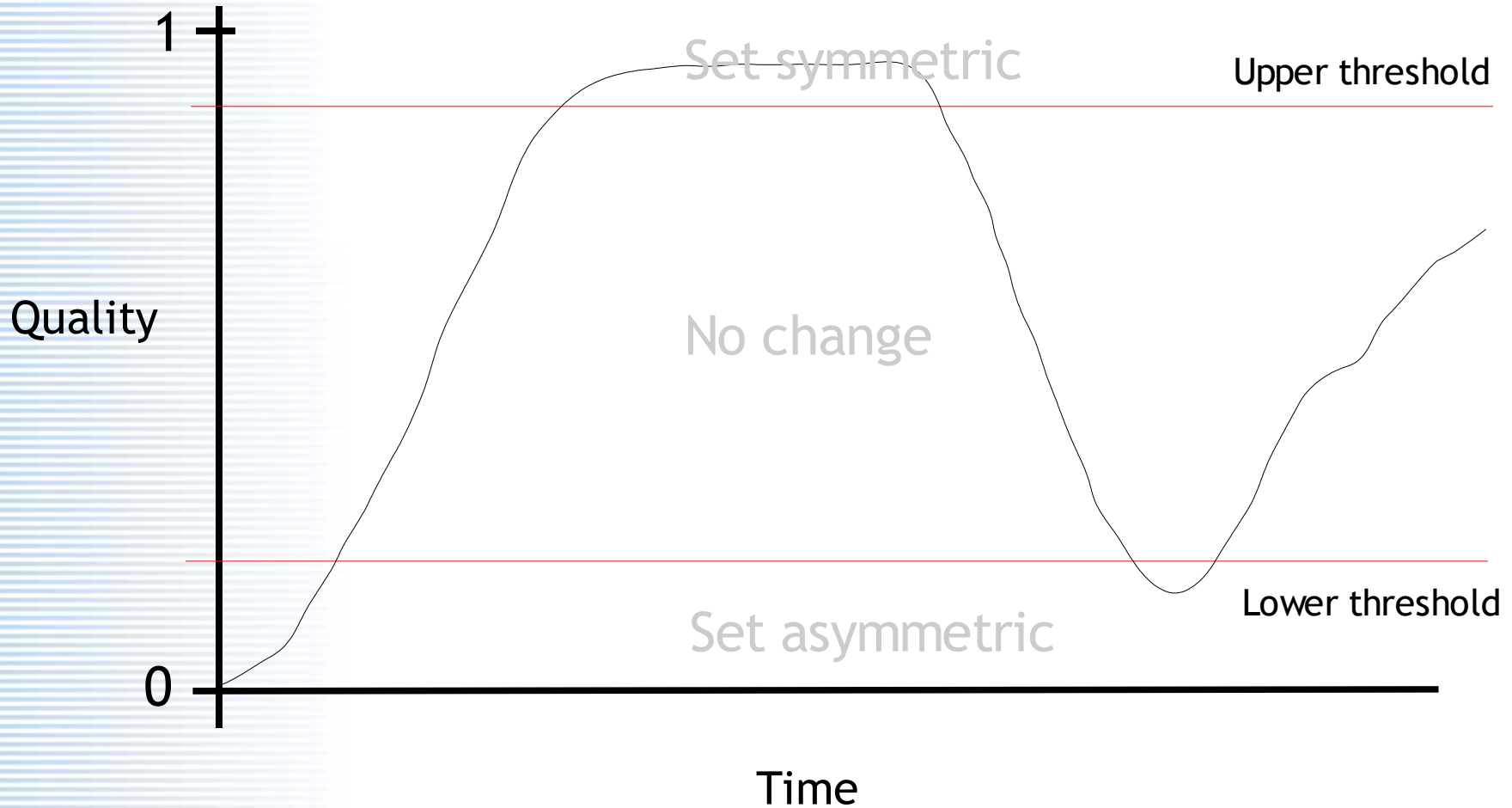
Link hysteresis is a more robust link-sensing scheme. It is used in addition to the core link-sensing.

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

Hysteresis maintains a *link quality* value for every detected neighbor. The calculation of this could be based upon link-layer information - but the RFC provides a algorithm that is quite “tuneable”.



Link hysteresis





Link layer notification

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

This will be link-status information - mainly to detect link-breakage.

This information should be used in link-sensing - both standard and link hysteresis.



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 badwidth 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>

- Background
- Challenges
- Basic structure
- Plug-ins
- Gateway tunneling



The project

Goal: Develop a RFC3626 compliant implementation and look into possible extensions (first goal: implement MID functionality in the INRIA OLSR daemon)

Based on draft3 compliant nolsrd1a10 by INRIA

Drafts went from version7 to 11 during spring -03

The last draft was version 11 - now experimental RFC(3626)

Almost all code rewritten - better to start from scratch?



Technical

Implementation for GNU/Linux systems(kernel >2.0)

Implemented in C

All areas of the RFC covered except link-layer notification

Supports IPv4 and Ipv6

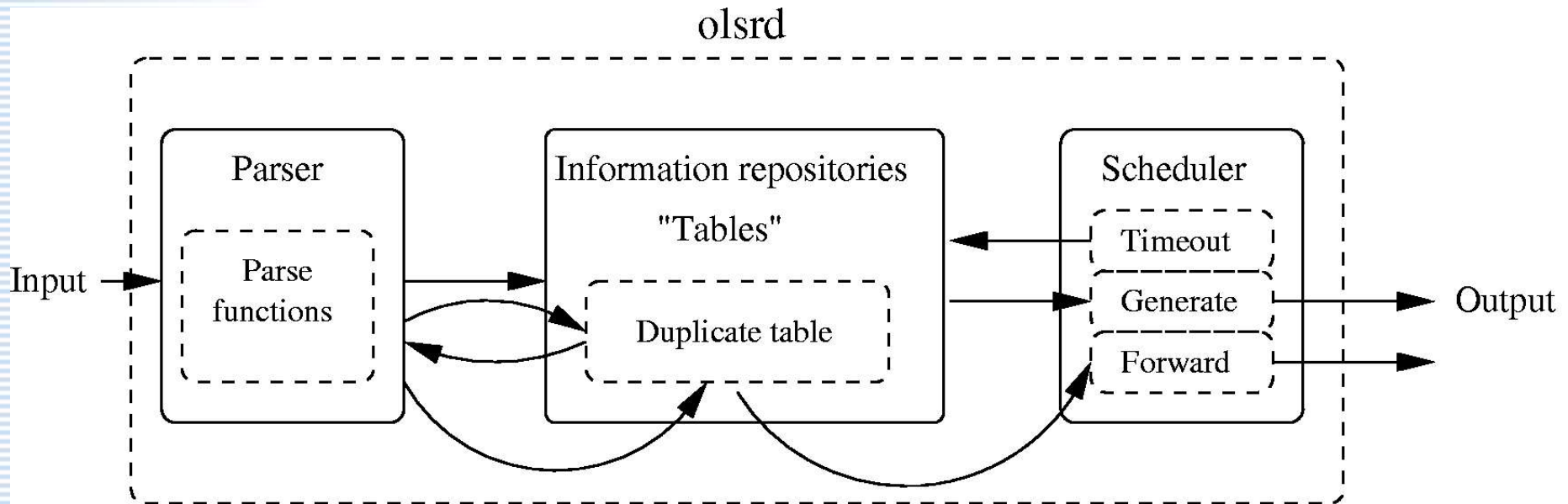
Licensed under the GeneralPublicLicense(GPL)

To get a feeling of the extent of the project a linecount of all code(daemon, GUI, plugins) returns 22861



Basic overview

All control traffic is broadcasted on UDP port 698



The scheduler runs in a thread of its own and data integrity is assured by using a *mutex* to protect the critical regions.



Basic overview

Parser - All incoming OLSR packets are chopped into messages by the parser. Functions parsing these messages registers with the parser at startup.

Scheduler - The scheduler runs scheduled tasks at given intervals. All table time-out functions and packet generation functions registers with the scheduler at startup.

The scheduler also handles triggered events like TC generation when MPR/neighborhood changes occur.

Both of these entities work on the different tables.



Some challenges

- Configuring interfaces and updating routes
- Binding sockets to devices
- Most possible transparent code considering Ipv4/6
- Fast searchable data storage structures
- Most possible modular design of parser/scheduler
- Implementing support for loadable plugins(DLLs) and designing a interface for DLL communication.
- Link layer notification



Current status

0.4.0 - latest release

- All core functioning covered
- All auxiliary functioning except link-layer

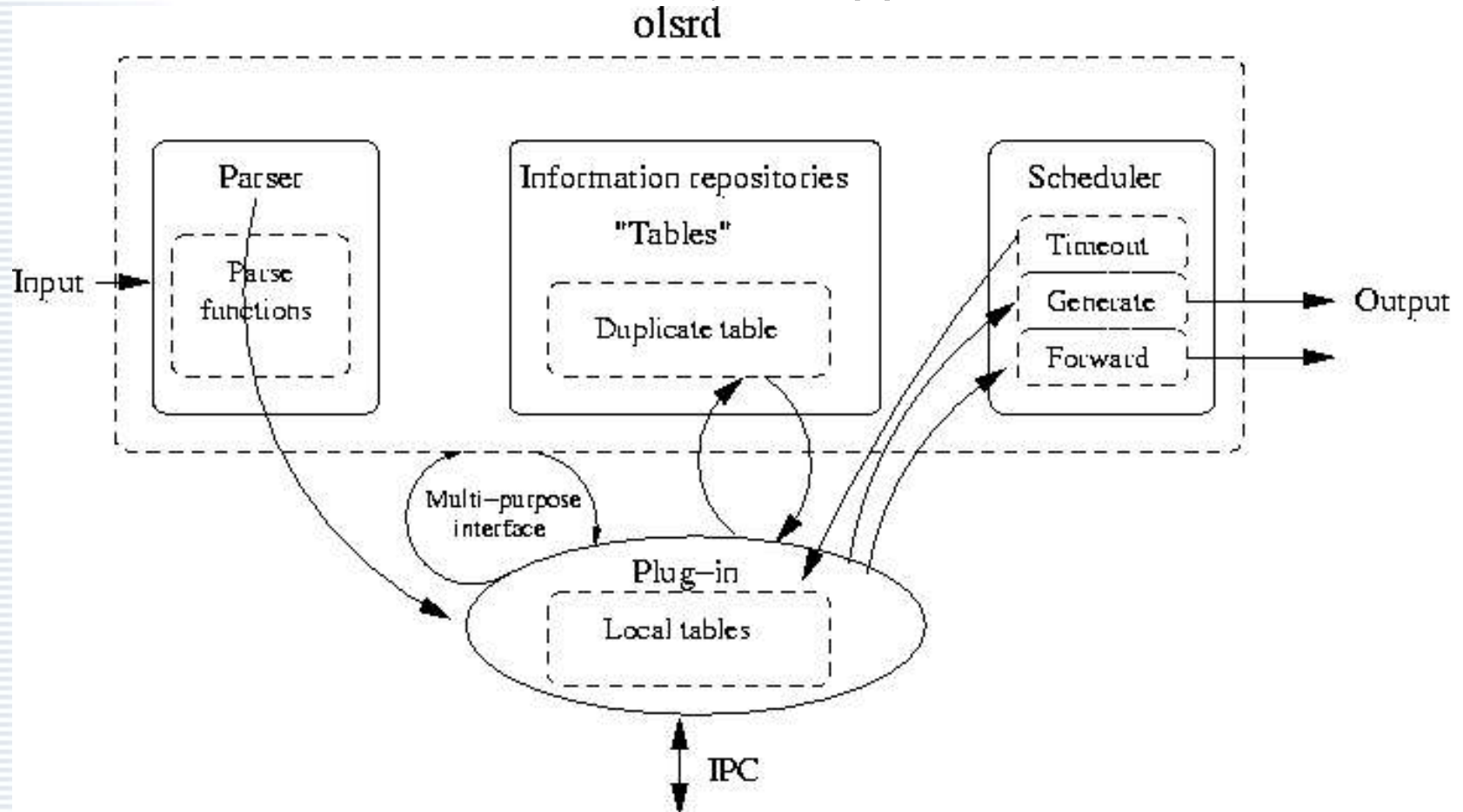
0.4.1 - next release

- Bugfixes
- Plugin support
- limited link-layer notification



Loadable plugins

A plugin(dynamically loaded library) is a piece of code that can be linked and “loaded” by an application in runtime.





Loadable plugins

- No need to change any code in the olsr daemon to add custom packages. For special needs some code additions may have to be done to olsrd.
- Users are free to implement olsrd plugins and licence them under whatever terms they like.
- If you, unlike yours truly, don't love C, the plug-ins can be written in any language that can be compiled as a dynamic library. Linux even allows scripts!
- No need for people with extended OLSR versions to rely on heavy patching to maintain functionality when new olsrd versions are released.



Link-layer notifications

- Linux WLAN drivers can “spy” on up to 8 nodes
- Nodes are registered using MAC addresses - ARP cache must be updated...
- Link info only updated on unicast traffic.



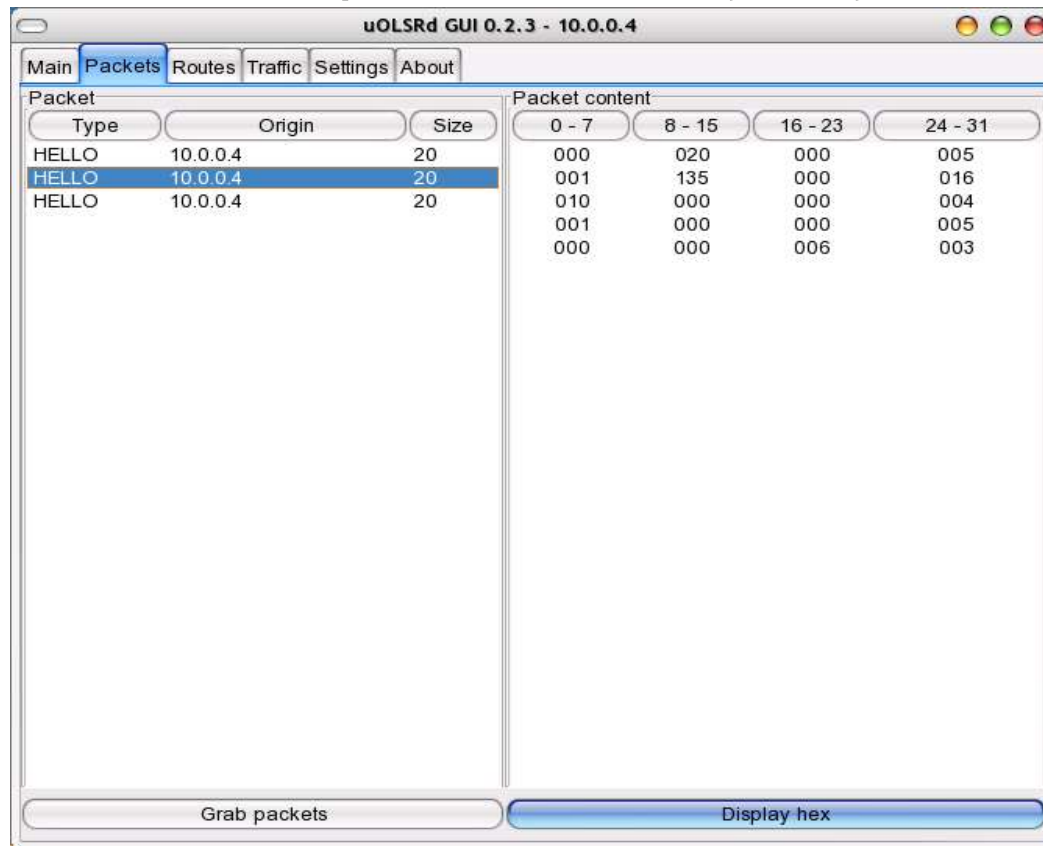
Other extensions

- GUI front-end
- Gateway tunneling
- Address auto configuration



A GUI front-end

A GUI front end is written using the GTK library
It communicates with the daemon over IPC and does not interfere with OLSR operation in any way.



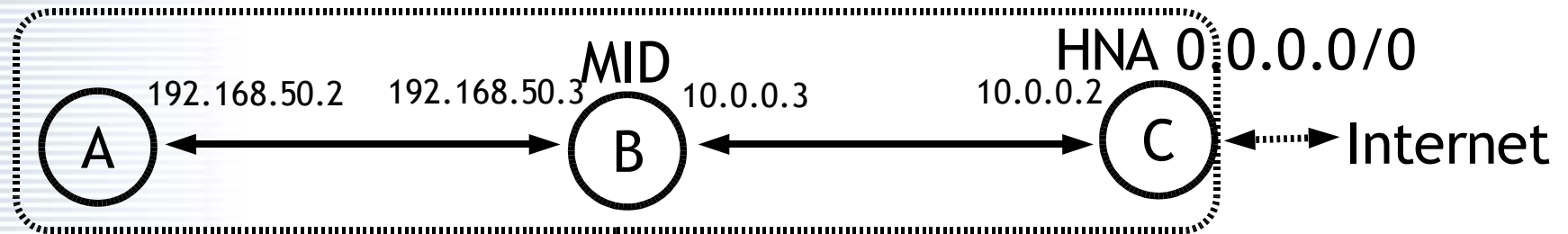


Gateway tunneling - background

As mentioned earlier a node has no actual control of what gateway it uses if multiple gateways for the same network exists.

This is due to the fact that the kernel routing requires you to have a route *to* your gateway.

Example - Node A wishes to set node C as its default gw:



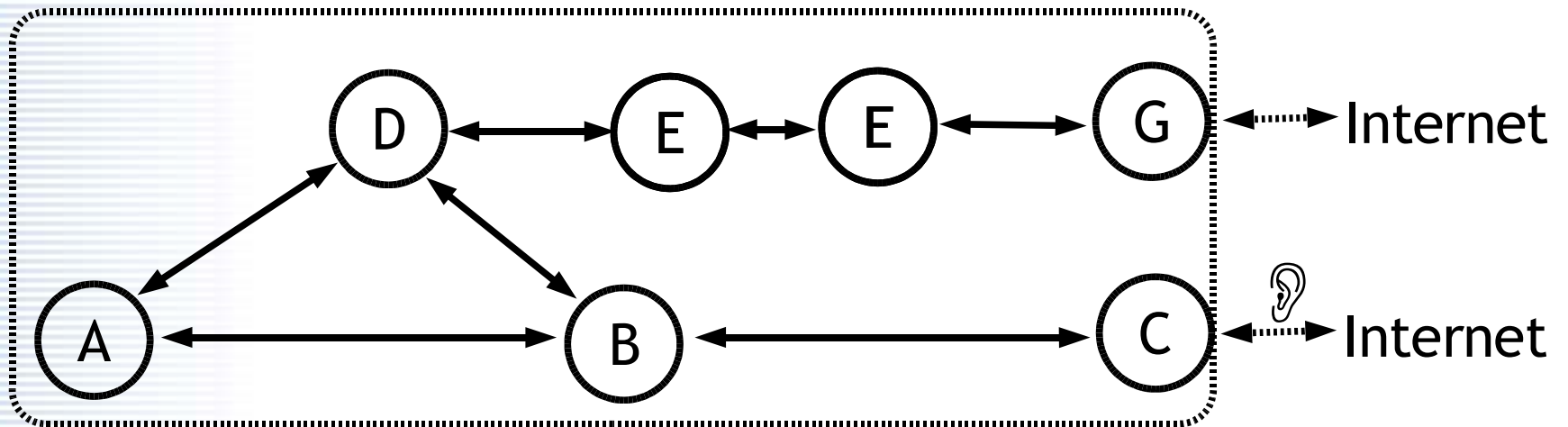
Node A has no route to the 10.0.0.0/24 network and is not allowed to add 10.0.0.2 as a gateway



Gateway tunneling - background

A has to set B as the default gateway and B again sets C as its default gateway - so the traffic is routed via C.

But what if A does not want to use C as a gateway?



Sending traffic to either D or B causes C to be used as gateway



Gateway tunneling

Motivation:

- Avoid possible TCP session breakage
- Security
- Load balancing

Solution: Node A sets up A IP-in-IP tunnel to the desired gateway and uses it regardless of hop-count.

