Better Approach To Mobile Adhoc Networking

batman-adv - Kernel Space L2 Mesh Routing

Martin Hundebøll

Aalborg University, Denmark

March 21st, 2014

History of batman-adv

The B.A.T.M.A.N. protocol initiated in Berlin, 2006. The first edition was developed as a daemon, and moved to kernel space in 2007 to improve performance.

Keywords:

- L2 routing (MAC addresses)
- Agnostic to IP or any L3 protocol
- Developed as a replacement/alternative to Open Link State Routing
- Runs on any Ethernet capable device (e.g. 802.3, 802.11, and 802.15.1)
- Supports client roaming
- Encapsulates incoming ether frames and handles all forwarding/delivery.
- Available by default in most Linux distributions (`modprobe batman_adv`)

Current development is focused on the next generation of the protocol, version 5, which is what I consider in this presentation.

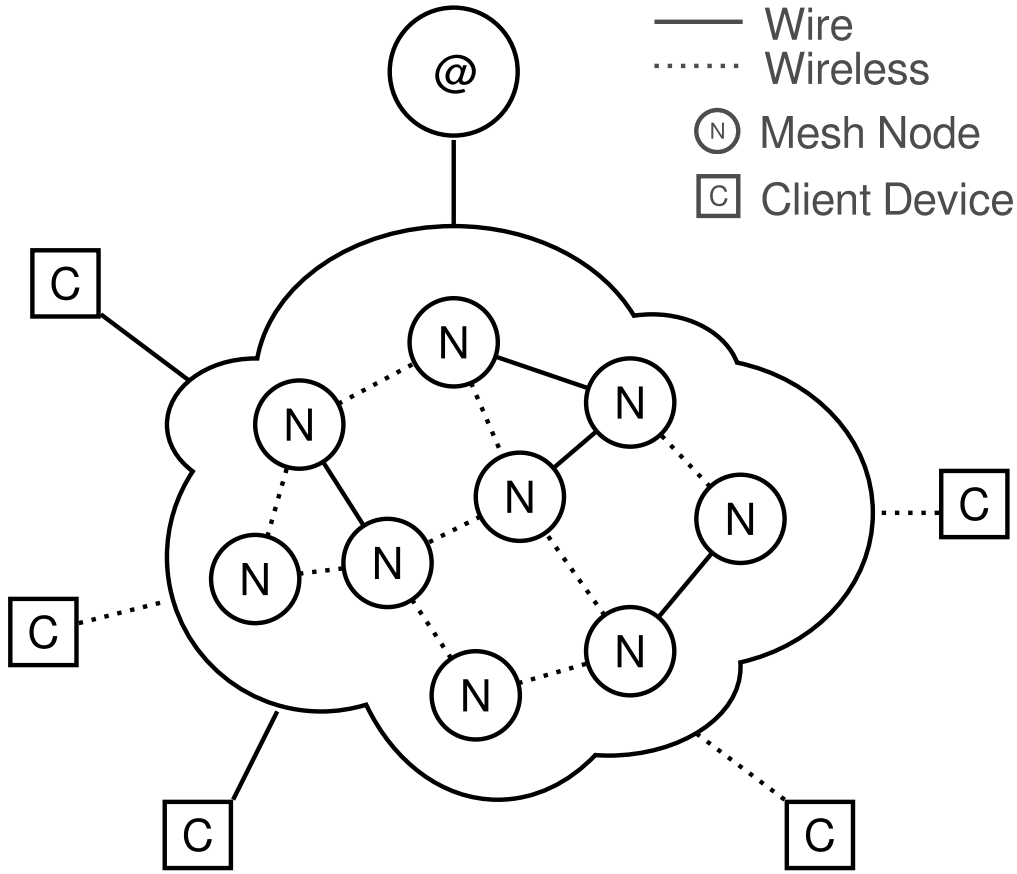
Features of batman-adv

During the presentation, the following features are discussed:

- Link Estimation
- Route Propagation
- Client Announcement
- Client Roaming
- Distributed ARP Table
- Bridge Loop Avoidance
- Multi-Link Optimizations
- AP Isolation
- Fragmentation
- Multicast Optimizations
- VLAN Support
- Information Distribution

Concept

A batman-adv mesh network can be considered as a big virtual Ethernet switch with outlets at every node in the network:



Common Setup

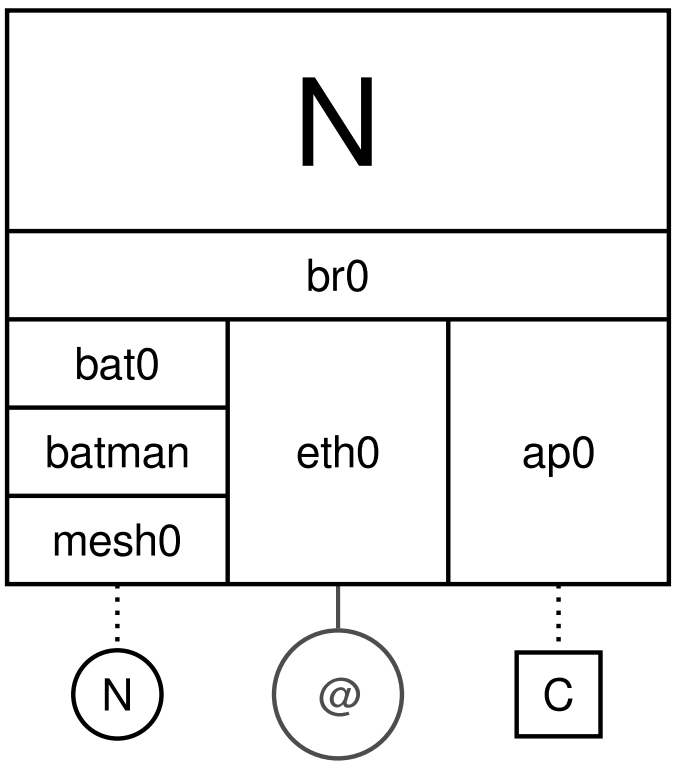
The usual mesh setup used by communities in Europe, South America, and more includes

- mesh backbone consisting of mesh nodes
- mesh clients that can roam between mesh nodes
- mesh gateways connected to the internet

The hardware is mostly consumer grade routers running openwrt.

Node Configuration

The mesh backbone is run on primarily ad-hoc wireless networks, while the clients are connected through access points running on the mesh nodes:



mesh0 is the ad-hoc interface

```
iw phy0 interface add mesh0 type ibss
```

bat0 is the entry to the mesh:

```
ip link add dev bat0 type batadv
ip link set mesh0 master bat0
```

ap0 is the access point provided to clients:

```
iw phy0 interface add ap0 type managed
hostapd /etc/hostapd/hostapd.conf
```

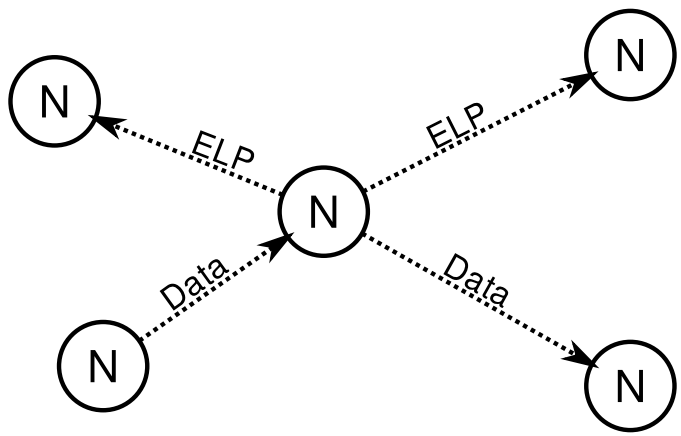
br0 is the bridge connecting them all:

```
ip add br0 type bridge
ip link set bat0 master br0
ip link set eth0 master br0
ip link set ap0 master br0
```

Neighbor Discovery and Link Estimation

Every node estimates one-hop link qualities towards every visible neighbor.

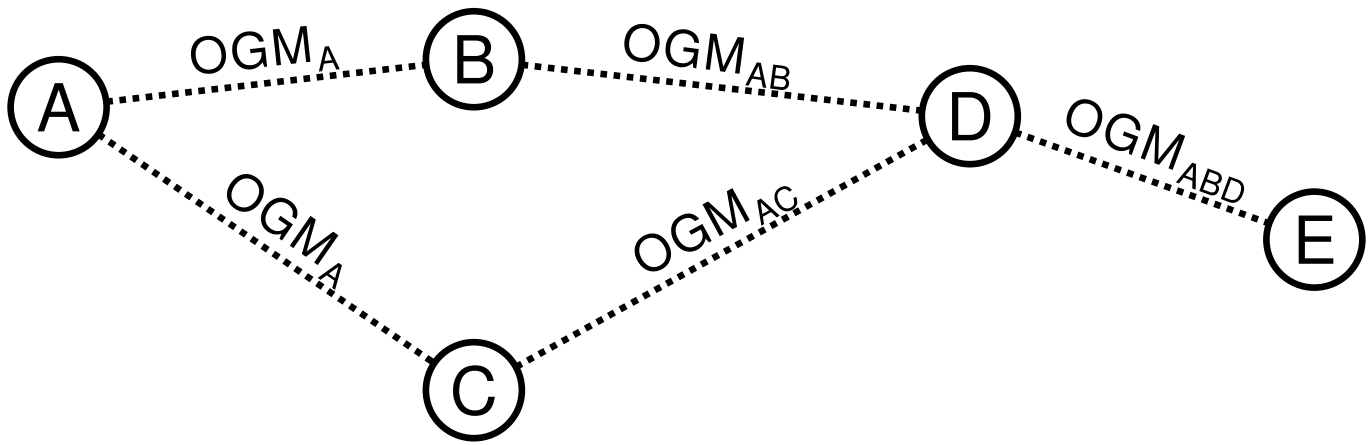
- ELP messages are broadcasted periodically to allow neighbor detection.
- To estimate the quality of a link, batman-adv queries the rate control algorithm of the wireless subsystem, which already keeps track of the estimated throughput.
- In case of an idle link, batman-adv transmits dummy frames (ELP messages) to trigger updates in the rate control system.
- To detect changes faster, ELP messages can be transmitted at a small interval (these are not forwarded by other nodes).



Beacons/Originator Messages

batman-adv uses Origator Messages (OGMs) to propagate routes in the network:

- Transmitted on a fixed interval (with some random jitter)
- Contains information about the originator of the message
- Carries the accumulated link quality estimate
- Nodes forward (aggregated) OGMs from the best route towards the originating node.

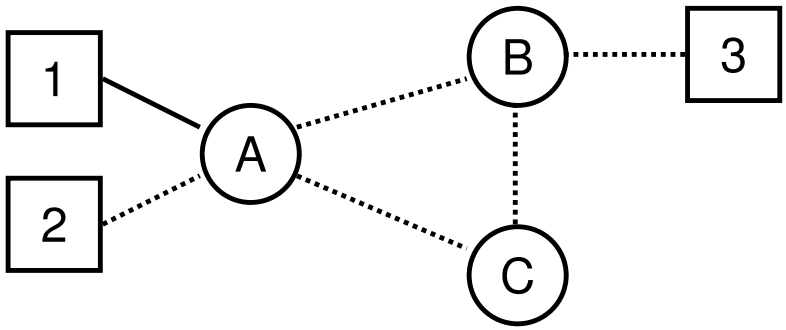


The link quality estimate from ELP/rate control is used to update the accumulated link quality in the OGMs.

Client Support

To serve clients connected through the bridge interface, batman-adv keeps track of the MAC addresses passing through `bat0`.

- Each node maintains a list of clients connected locally
- Changes to the list are piggybacked on OGMs
- Each node maintains a list of every client connected globally.
- For every global client, the list tells which node is serving the client.



Node A	
local	global
1	
2	
	3 ⇨C

Node B	
local	global
	1 ⇨A
	2 ⇨A
3	

Node C	
local	global
	1 ⇨A
	2 ⇨A
	3 ⇨B

When frames arrives through `bat0`, the node can lookup the node serving the destination client.

Client Roaming

When clients roam from the access point of one node, to the access point of another node, the following (simplified) steps happens:

1. The new serving node sees a client from the global list
2. The new node notifies the previous serving node.
3. The previous node now reroutes the client-traffic to the new node.
4. The both nodes advertises the change within the next OGM message.

In case notifications or changes are lost due to packet errors, the system detects inconsistencies by version numbers and checksums and recover by requesting full tables from involved nodes.

Distributed ARP Table

When using IPv4 on top of batman-adv, clients resolves addresses with ARP. Requests are broadcasted, and thus vulnerable to packet loss, which leads to ARP timeouts.
batman-adv features a Distributed ARP Table that snoops ARP messages and caches the responses in the mesh network.

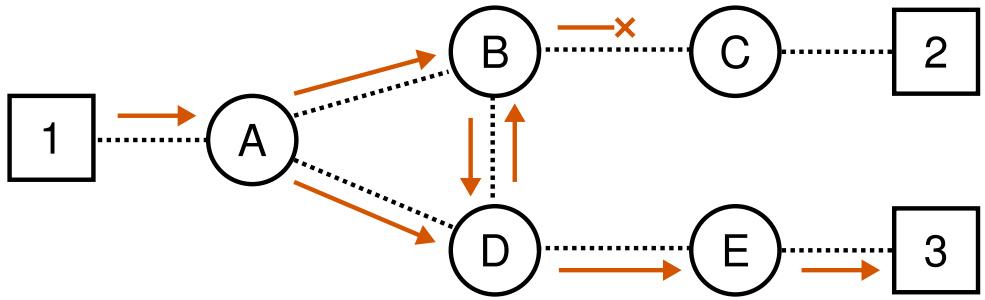


Figure : Client 1 broadcasts an ARP request to resolve Client 2. In this example the request is lost after Node B.

DAT works like this:

- batman-adv solves this by caching ARP entries in a *Distributed Hash Table* (DHT), where each node caches a subset of the entries.
- ARP requests from clients (1) are snooped at the serving node (A), who looks up candidate DHT nodes (E).
- A request is transmitted with unicast to the candidate nodes (E), who responds with the resolved MAC address.
- The serving node (A) responds with an ARP reply and stores the entry for later ARP requests.

Distributed ARP Table

When using IPv4 on top of batman-adv, clients resolves addresses with ARP. Requests are broadcasted, and thus vulnerable to packet loss, which leads to ARP timeouts.
batman-adv features a Distributed ARP Table that snoops ARP messages and caches the responses in the mesh network.

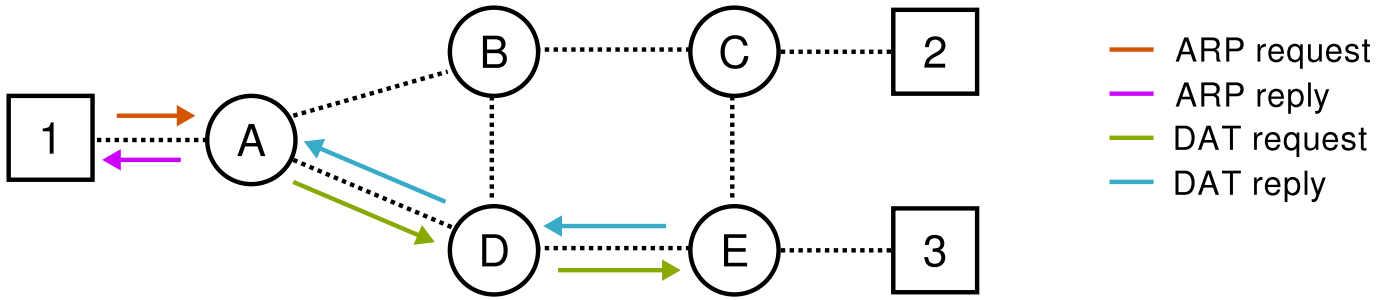


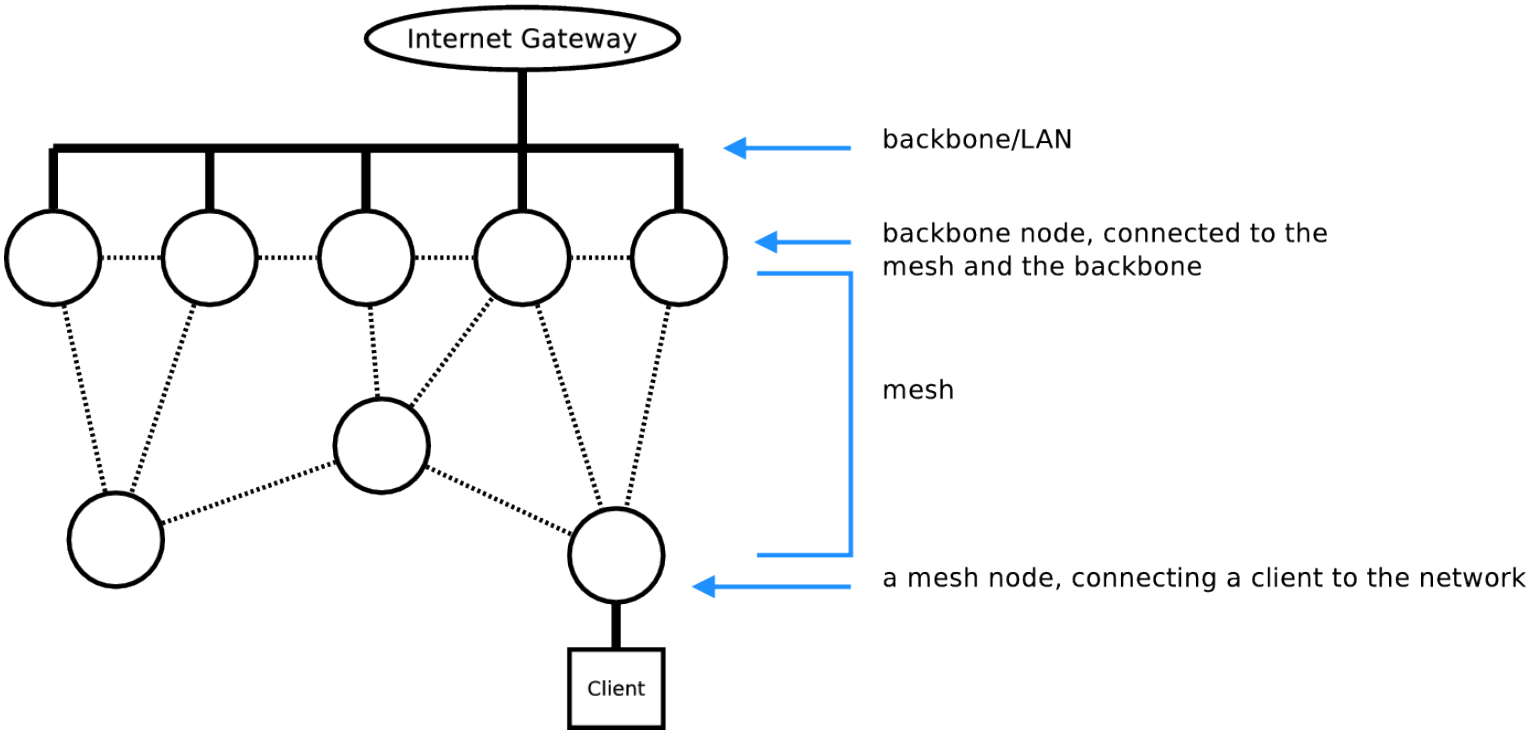
Figure : With DAT, the ARP request is intercepted and the entry is requested from Node E.

DAT works like this:

- batman-adv solves this by caching ARP entries in a *Distributed Hash Table* (DHT), where each node caches a subset of the entries.
- ARP requests from clients (1) are snooped at the serving node (A), who looks up candidate DHT nodes (E).
- A request is transmitted with unicast to the candidate nodes (E), who responds with the resolved MAC address.
- The serving node (A) responds with an ARP reply and stores the entry for later ARP requests.

Bridge Loop Avoidance

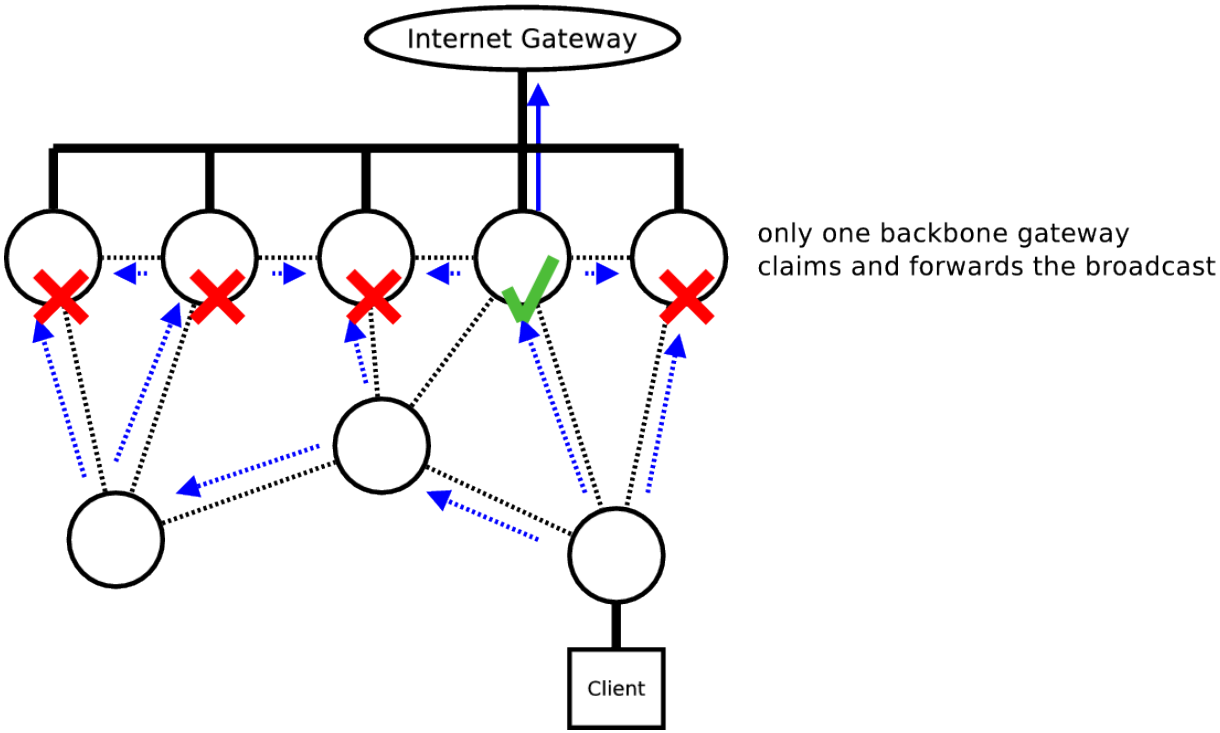
In cases where multiple gateway from the mesh are connected to the same backbone, broadcast storms can occur.



Bridge Loop Avoidance handles this by claiming clients in the mesh network.

Bridge Loop Avoidance

In cases where multiple gateway from the mesh are connected to the same backbone, broadcast storms can occur.

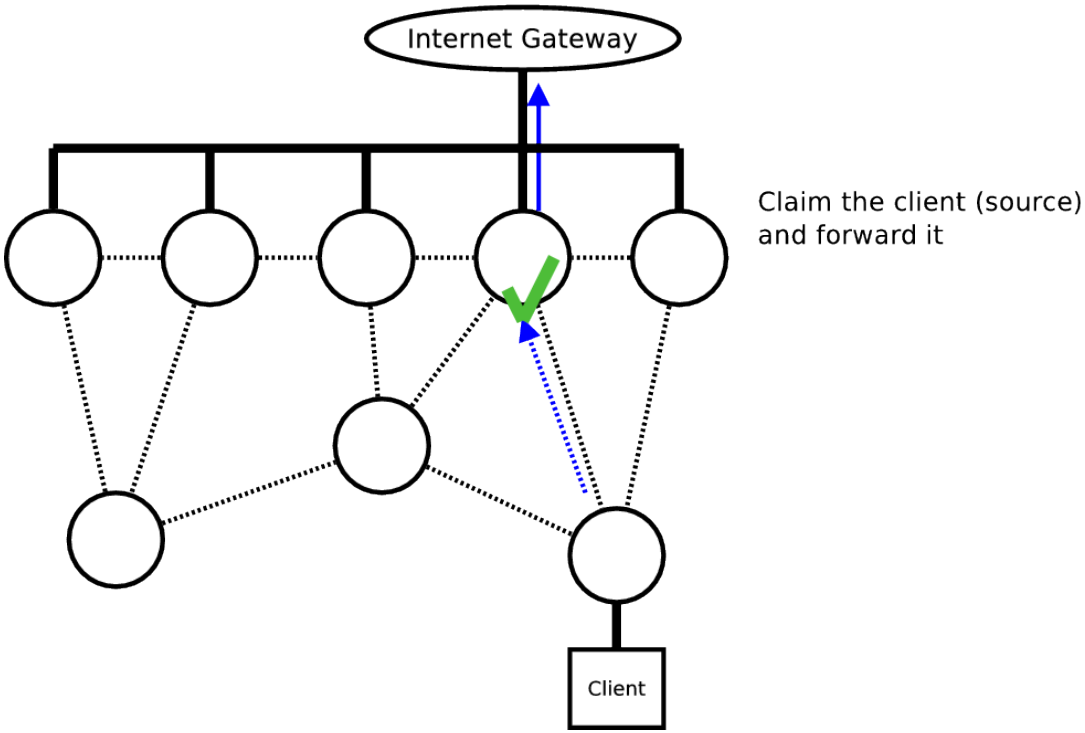


Bridge Loop Avoidance handles this by claiming clients in the mesh network.

- A broadcast from an unclaimed client is claimed by the first receiving gateway. Other gateways drops broadcasts from the claimed client.

Bridge Loop Avoidance

In cases where multiple gateway from the mesh are connected to the same backbone, broadcast storms can occur.

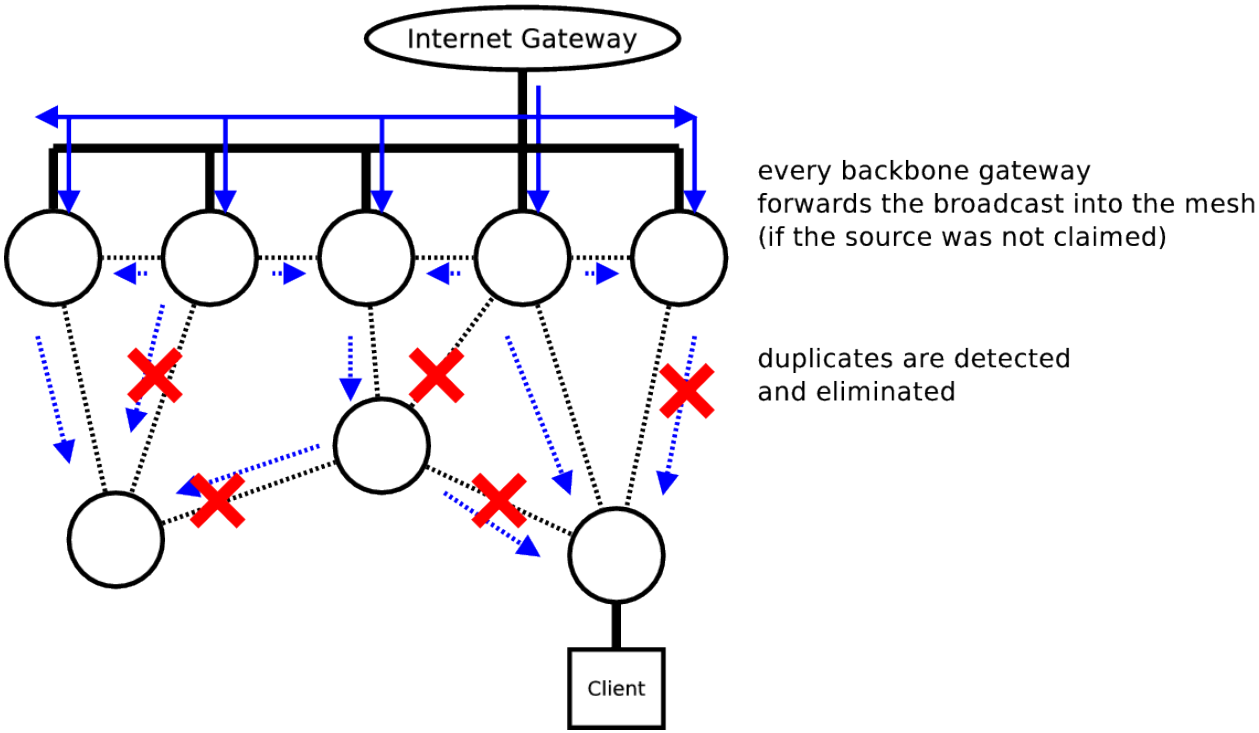


Bridge Loop Avoidance handles this by claiming clients in the mesh network.

- Clients can also be claimed whenever unicast traffic from the client is bridged by the gateway.

Bridge Loop Avoidance

In cases where multiple gateway from the mesh are connected to the same backbone, broadcast storms can occur.

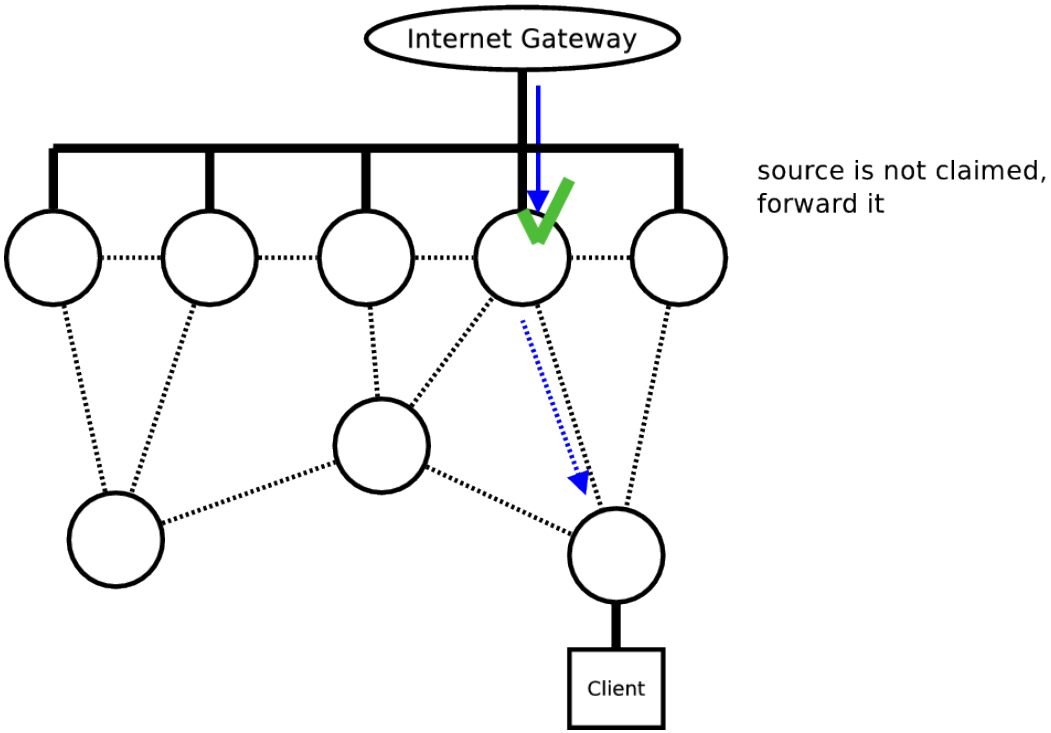


Bridge Loop Avoidance handles this by claiming clients in the mesh network.

- Broadcasts from the backbone are forwarded by all gateways, and duplicates are eliminated in the mesh with checksumming.

Bridge Loop Avoidance

In cases where multiple gateway from the mesh are connected to the same backbone, broadcast storms can occur.

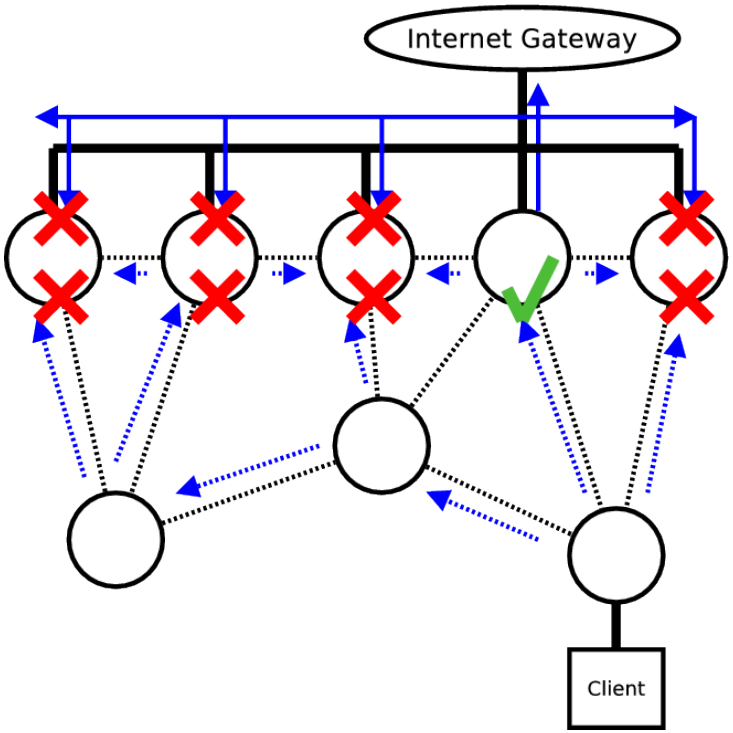


Bridge Loop Avoidance handles this by claiming clients in the mesh network.

- Switches send unicast traffic to one gateway only, in which case the packet is simply forwarded into the mesh.

Bridge Loop Avoidance

In cases where multiple gateway from the mesh are connected to the same backbone, broadcast storms can occur.



the other backbone gateways don't forward the broadcast back into the mesh because the source is claimed

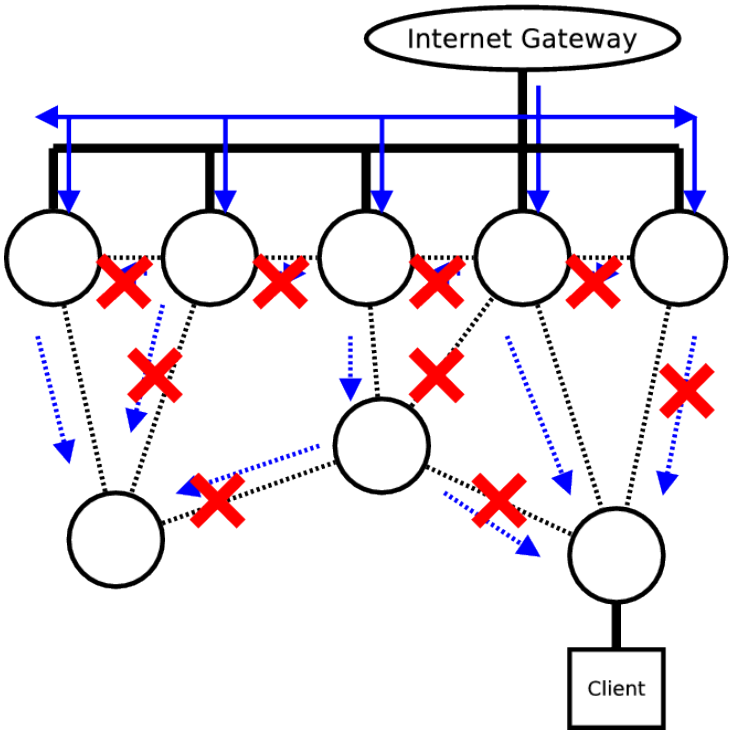
only one backbone gateway claims and forwards the broadcast

Bridge Loop Avoidance handles this by claiming clients in the mesh network.

- When broadcasts are forwarded onto the backbone, other gateways drop the packet.

Bridge Loop Avoidance

In cases where multiple gateway from the mesh are connected to the same backbone, broadcast storms can occur.



every backbone gateway
forwards the broadcast into the mesh
(if the source was not claimed)

backbone gateways don't forward broadcasts
into the LAN originated by other backbone gateways

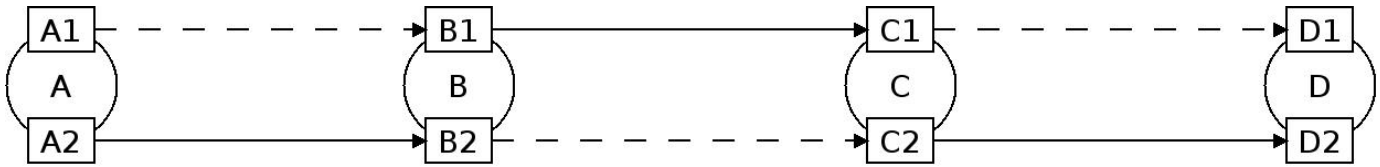
duplicates are detected
and eliminated

Bridge Loop Avoidance handles this by claiming clients in the mesh network.

- Same happens when gateways forwards broadcast packets onto the mesh. These are not forwarded back onto the backbone by other gateways.

Network Wide Multi Link Optimization

To exploit paths with alternating channels, batman-adv nodes broadcast distinct OGMs on each available. When forwarding OMGs from other nodes, a penalty is added when forwarding onto the interface, on which the OMG arrived.

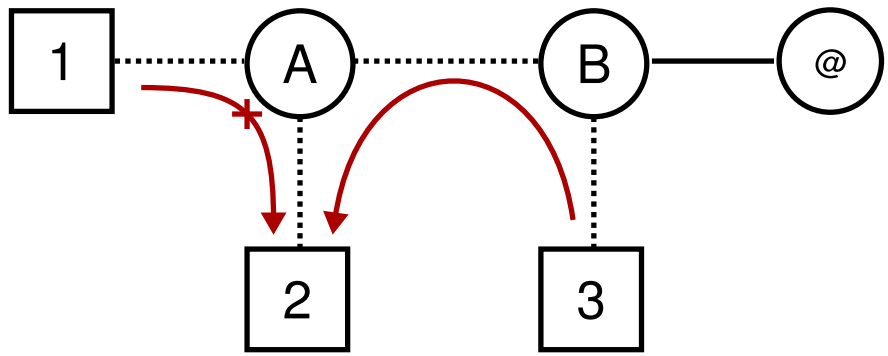


Example:

- Node A transmit an OGM on A1 and another on A2.
- Node B receives an OGM on B1, and forwards it on B2 with hop-penalty and on B1 with both hop-penalty **and** link-penalty.
- Node B received another OGM on B2. This is only forwarded on B1, as the previous OGM has the better metric due to link-penalty.
- The last two steps are repeated on Node C.
- When Node D transmits packets to Node A on D1, Node C will see a better path towards Node A on C2, and so on.

AP Isolation

With a single access point, clients can be isolated from one another by the means of *AP Isolation*. With batman-adv, multiple access points are used and thus AP isolation needs to be handled within batman-adv.

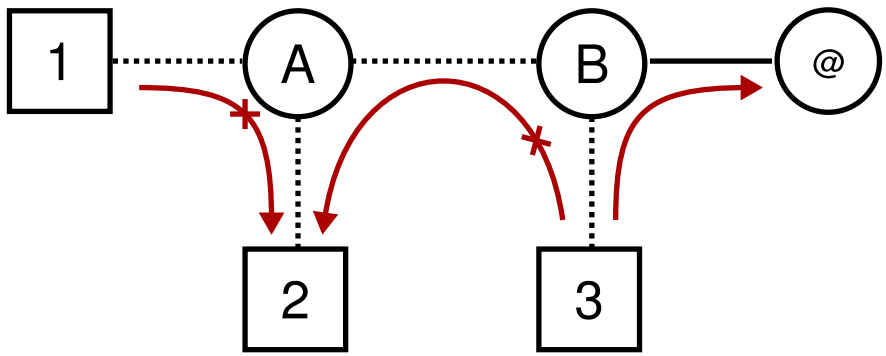


With AP isolation in batman-adv:

- Unicast traffic between isolated clients is dropped at the first node.
- Broadcast traffic from isolated nodes can be dropped at outgoing interfaces.

AP Isolation

With a single access point, clients can be isolated from one another by the means of *AP Isolation*. With batman-adv, multiple access points are used and thus AP isolation needs to be handled within batman-adv.



With AP isolation in batman-adv:

- Unicast traffic between isolated clients is dropped at the first node.
- Broadcast traffic from isolated nodes can be dropped at outgoing interfaces.

Fragmentation

Since batman-adv encapsulates entire ethernet frames, the size is increased with protocol data.

Figure : With increased MTU between Node A and Node B

When the MTU of the ad-hoc interfaces cannot be increased accordingly, fragmentation is needed.

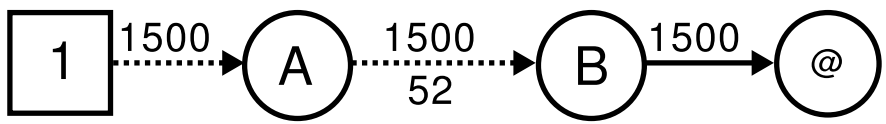


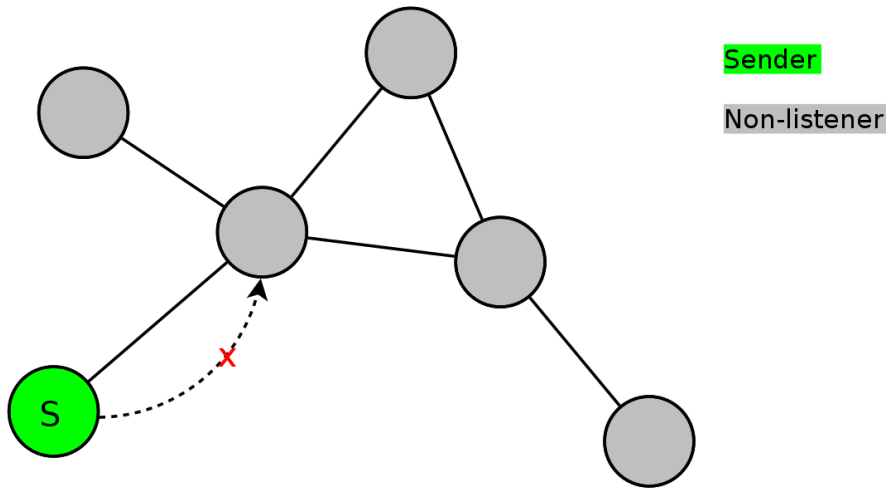
Figure : The MTU between Node A and Node B is too small, and the packet is fragmented.

With fragmentation, the encapsulated frame is transmitted in fragments, which are merged at the receiver. This is also used for information about clients in the network.

Mulicast Optimizations

batman-adv handles multicast traffic according to three different scenarios:

No listeners Multicast packets are dropped at the gateway.



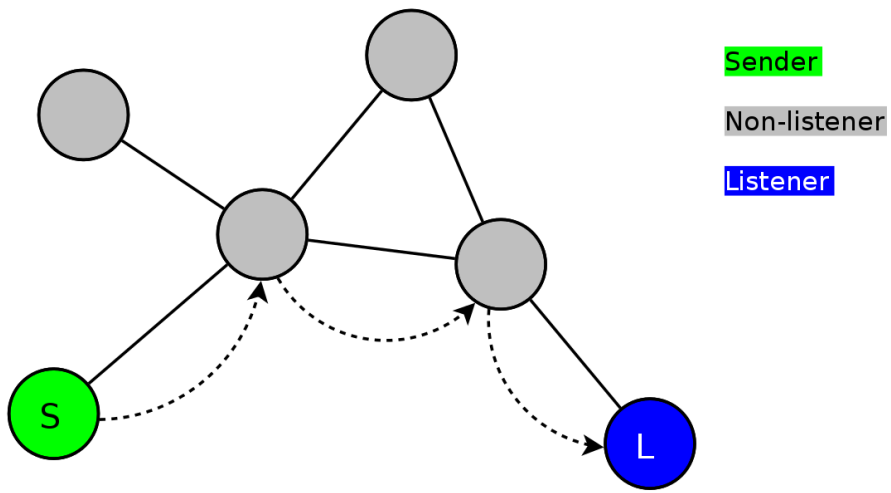
Field studies show that multicast groups often have zero or one listeners.

Mulicast Optimizations

batman-adv handles multicast traffic according to three different scenarios:

No listeners Multicast packets are dropped at the gateway.

Single listener Multicast packets are forwarded as unicast.



Field studies show that multicast groups often have zero or one listeners.

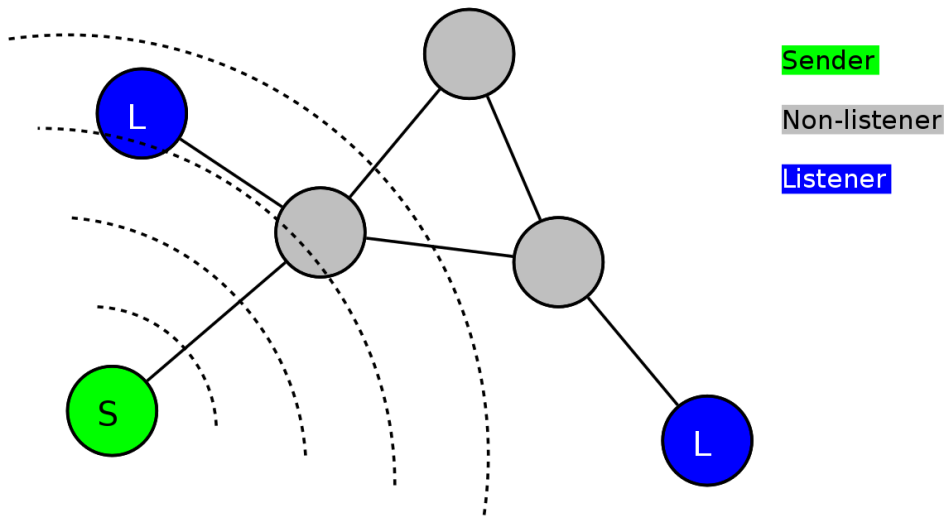
Mulicast Optimizations

batman-adv handles multicast traffic according to three different scenarios:

No listeners Multicast packets are dropped at the gateway.

Single listener Multicast packets are forwarded as unicast.

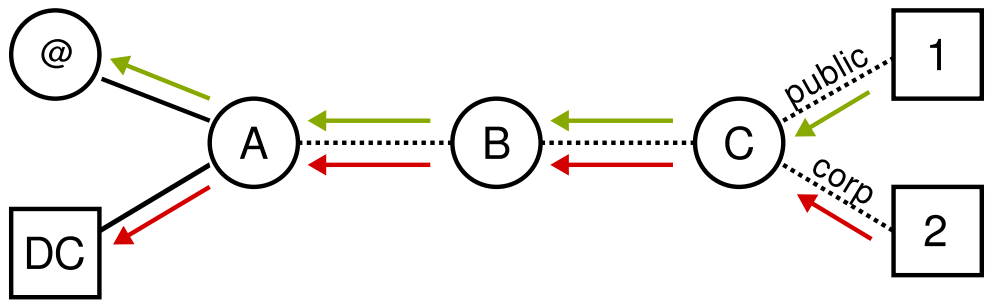
Multiple listeners Multicast packets are flooded as broadcast.



Field studies show that multicast groups often have zero or one listeners.

Virtual LAN

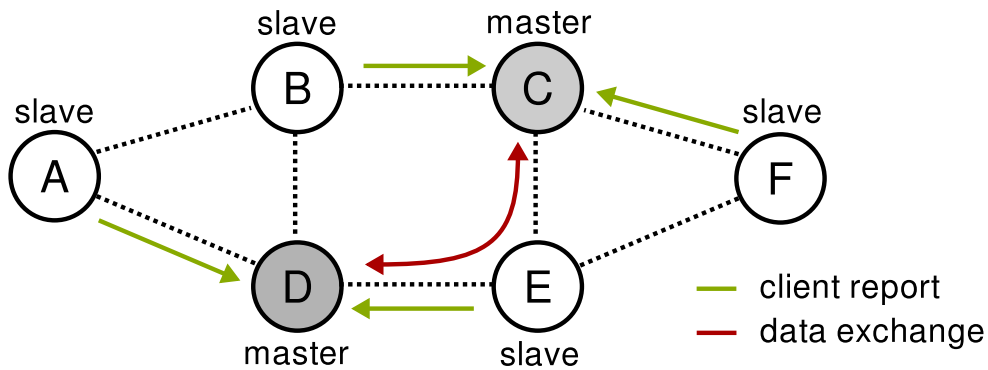
batman-adv supports VLAN throughout the mesh, meaning clients are separated in two distinct LANs, e.g. public internet access, and corporate intranet access.



Many settings in batman-adv are supported on a per-VLAN-basis. E.g. having public clients isolated, while corporate clients can communicate.

Distributed Network Information

Often one wishes to collect data from the network, e.g. network links, hostnames, gps locations, etc. To avoid having every node broadcasting all the information to all nodes, A.L.F.R.E.D. supports distributed collecting of information.



Information flow:

- Information is locally passed to slaves through a UNIX socket.
- Slaves reports information to the closest master (using info from batman-adv).
- Masters synchronize information with each other.
- Any slave or master can request the full information from a master.

The number of masters is up the administrator.