InfiNet Wireless

MINT handbook

Updated**: 19 August 2009**

Legal Rights

Statement of Conditions

Disclaimer

InfiNet Wireless hereby declares that R5000-Om, R5000-Mm, R5000-Sm and R5000-Lm are in compliance with the essential requirements and other relevant provisions of Directive 1995/5/EC.”The declaration of conformity may be consulted at http://www.infinetwireless.com/products-technologies/type-approval-certificates/DoC_RTTE.pdf.

Indication of the countries

InfiNet Wireless equipment has no geographical limitations for selling and can be supplied to any country of the world.

Limitation of Liability

International Regulatory Information

This is a class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures. Hereby, InfiNet Wireless declares that this equipment is in compliance with the essential requirements and other relevant provisions of Directive 1999/5/EC.

CE	**The CE Marking on this equipment indicates compliance with the following:** **This device conforms to Directive 1999/5/EC on Radio Equipment and Telecommunications Terminal Equipment as adopted by the European Parliament And Of The Council.**

Table of contents

I. MINT GENERAL DESCRIPTION 5

II. SWITCHING IN MINT NETWORK 10

1. General description 10
2. Trunk groups 11
3. Studying example 11
4. Video solution switch configuration. 15
5. Management by means of switching 17

III. ROUTING IN MINT NETWORK 18

IV. ADDITIONAL MINT CAPABILITIES 19

1. Join 19
2. Prf (Pseudo radio interface) 20

V. TOPOLOGY MANAGEMENT 21

VI. ROAMING 24

I. MINT general description

MINT stands for Mesh Interconnection Network Technology which points to the technology for networks based on arbitrary connections. Throughout this document lots of details will be revealed and this will be coupled with real-life examples.

The most important feature of MINT architecture lies in its ability to present any wireless (or even sometimes wired) network as a flat Ethernet segment, and radio interface connected to this network will act as usual Ethernet interface (virtual). By saying “virtual” this means that it has its own specific settings and parameters such as frequency, modulation type etc. But applications which work with this interface will not “know” about it and will not tell the difference between this interface and other Ethernet interfaces.

Let’s get started and look at the examples described below.

Let’s imagine we have two InfiNet Wireless units each of which has one “real” Ethernet interface and one radio interface (although some modifications might have more than one radio interface).

No explanations needed for “real” Ethernet – so let’s try to configure the radio.
In order to make two radios communicate with each other using radio interface we should configure them so that the units could physically “talk” to each other. The simplest way to do that is to configure the same parameters for radio interfaces of both units.

For example:

```
rf rf4.0 freq 5200 bitr 36000 sid 10101010 pwr 63
```

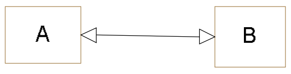
Here, we have configured our radio interface for the following parameters:

5200 – frequency 5200 MHz
Bitrate – 36 Mbps
SID – 10101010
Output power 63 mW (18 dBm)

Then, execute the “magical” command on both units which will start MINT protocol:

```
mint rf4.0 start
```

And… what do we see? Nothing?
In fact, our units have already exchanged a series of service packets and have established the connection, have agreed on the optimal parameters for data transmitting and are ready for work.
Connection establishing takes less than a second.



To check our current list of connections (look up the list of this unit’s neighbors) one can use the following command:

```
mint map
==========================================================
Interface rf4.0, node 00028AE1DAC1 "" id:32456 (mesh)

 1 Neighbors:
 ------------
  33521            001195FD8FFB, Cost=30 , I/O=15/18 <36/36> /mesh/
==========================================================
```

What do we learn from this table?
The first line shows the name of the radio interface on which our MINT protocol is running (rf4.0), unit's MAC-address for rf4.0 radio interface, our unit's identifier (32456) and its type (mesh).

Below we see that our unit (in MINT's terminology – node) has one neighbor. A neighbor has its own MAC-address and identifier.

These parameters are followed by **Cost** which is a very important parameter in MINT used for choosing optimal paths in a network with multiple nodes and connections. Its physical meaning – an estimated time for packet delivery measured in conventional units. The less the Cost, the higher probability that this path will be chosen. The Cost of each connection is constantly changing according to link parameters including energetic parameters (signal levels), modulation types, number of errors and retries, link load and other parameters thus allowing quickly switching to an alternative route if its cost will be lower than for the current one.
Of course, in our first example for point-to-point connection this parameter does not play any serious role but we will definitely speak about it further on.

Next parameters we observe are SNR (Signal/Noise Ratio) of sent and received signal in dB in decibels and current TX/RX bitrates for this particular neighbor chosen for current moment (or strictly fixed by the administrator).

Last parameter shown is neighbor's list of additional properties. Here, we see that our neighbor node is working in '**mesh**' node (see explanation below).

Node identifier, additional numeric parameter, can be configured by the administrator for better representation of a neighbors table. Lists will be sorted according to this identifier. The parameter has no specific importance for the protocol. You may just enumerate all nodes in ascending manner or make up any other numbering system for your convenience. If this parameter is not set by the administrator, a serial number of the unit will be set as a node identifier.

In our next example, let's assign node identifiers and node names for our two units.

First node will have an identifier '10' and a name 'Node_10':

```
mint rf4.0 –nodeid 10
mint rf4.0 –name "Node_10"
```

Node 2: '20' as identifier, 'Node_20' as a name:

```
mint rf4.0 –nodeid 20
mint rf4.0 –name "Node_20"
```

After these manipulations, our neighbors table looks as follows:

```
mint map
=============================================================
Interface rf4.0, node 00028AE1DAC1 "Node_10" id:10 (mesh)

 1 Neighbors:
 ------------
  00020 Node_20       001195FD8FFB, Cost=30 , I/O=15/18 <36/36> /mesh/
=============================================================
```

Now we have the connection but how can we use it?
The simplest way to test the connection is to assign IP-addresses to unit's interfaces and use ping to test the connectivity.

For example, for the first unit we configure IP-address:

```
ifconfig rf4.0 10.0.0.1/24 up
```

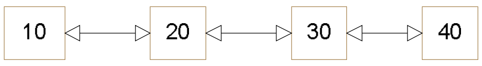
Second unit:

```
ifconfig rf4.0 10.0.0.2/24 up
```

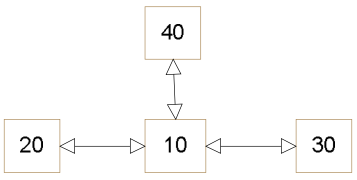
After this you can test the connection using ping. Even at this step you can utilize this link by configuring routing between eth0 and rf4.0 interfaces (suppose that it is evident how to do that using static routing or dynamic routing by means of RIP or OSPF protocols).

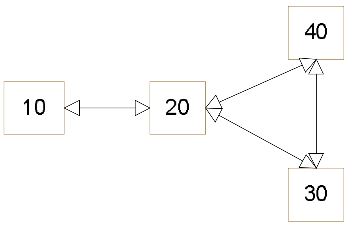
What will be our next step in MINT exploration?
Let's add two more nodes to our network.
Radio interfaces will be configured in the same way as we did that for the first two units. For MINT interface, let's assign '30' and '40' as identifiers and IP-addresses 10.0.0.3/24 and 10.0.0.4/24 correspondingly.

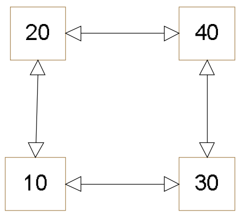
As we have not done any specific manipulations to our network's topology and all our nodes are working in '**mesh**' mode (by default) our network might look in many different ways in terms of connection between the units:



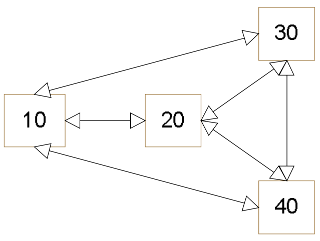
or

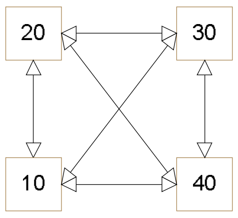






or





In the list of neighbors for the node ‘10’ we can potentially see one, two or three neighbor nodes. Regardless the topology, packets from node ‘10’ to node ‘40’ will be easily delivered. But what path on the network will be chosen?
For example, in the last picture for the topology there are 5 alternative paths between node ‘10’ and node ‘40’:

10->40
10->20->40
10->30->40
10->20->30->40
10->30->20->40

Which path is the optimal one?

The first one? Not necessarily.

It may turn out that the most optimal way (the fastest) will be the last one (the longest one in terms of the number of hops).
It all depends upon each links’ current parameters and conditions.
Car drivers surely know that the shortest path is not always the fastest one.
So, here ‘**Cost**’ parameter comes into picture.
MINT calculates all possible routes and chooses the most optimal one judging by the link costs sum for each possible route. And this is a continuous process and MINT will recalculate costs in case of any topology or link conditions changes to ensure that the route between any two network nodes is optimal. Moreover, MINT is able to forecast the situation by switching to alternative routes in case if the situation with active route tends to get worse. Switching between routes takes such a little time that this does not lead for upper layers protocols to break their connections.
In fact, there is no need to recalculate all the paths – this would become too time- and resource consuming. Each node “trusts” its neighbors which have already made some of the calculations and chooses the neighbor which offers an optimal path to the target node. Therefore, the task is to correctly evaluate the cost to the direct neighbors and use it along with routes which they are offering.

Full list of all neighbors can be printed using the following command:

```
mint map full
=====================================================
Interface rf4.0, node 00028AE1DAC1 "Node_10" id:10 (mesh)

 2 Neighbors:
------------
  00020 Node_20         001195FD8FFB, Cost=40 , I/O=24/27 <36/36> /mesh/
  00030 Node_30         000435FFC283, Cost=110, I/O=14/12 <24/24> /mesh/

 4 Routes:
---------
 00010: 000435FF9359 --> 00000: 00028AE1DAC1 Cost=0     Zone=0
 00020: 001195FD8FFB --> 00020: 001195FD8FFB Cost=40    Zone=1
 00030: 000435FFC283 --> 00030: 000435FFC283 Cost=110   Zone=1
 00040: 000435FFB65B --> 00020: 000435FF9357 Cost=200   Zone=2
=====================================================
```

From this table one can see that nodes 20 and 30 are our direct neighbors, node 40 is reachable via Node_20 and has a total cost of the path of 200 and is two hops away. Direct neighbors have a distance of one hop.

Important to remember, that every node independently calculates the cost of connections to its neighbors and this means that direct and reverse paths between two nodes may be different. Moreover, even direct neighbors may not choose a one-hop path but use other multi-hop paths if it turns out that those paths have less cost.

II. Switching in MINT network

1. General description

Now, having got a basic understanding on what MINT technology is, let's review another important feature of new architecture – switching.

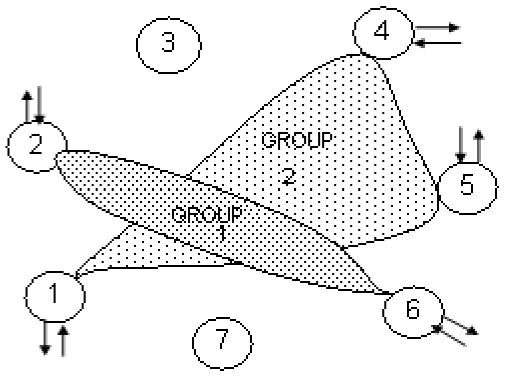
Traditionally, InfiNet Wireless units were only IP routers. Having kept routing capabilities, we have added switching functions which allow sending all or part of the traffic by means of switching. Switching may be used for sending data of non-routed protocols or just for ease of configuration and operation.
Both routing and switching may be active simultaneously.

To begin with, a little bit of theory.

Switch works over any network interfaces Ethernet 802.3 which support addressing and headers in Ethernet II (DIX) and 802.2 formats including virtual Ethernet formed by MINT technology.

Switch configuration is a set of rules for switching groups. Each group has a unique numeric identifier (1-4094).

Each switching group includes two or more network Ethernet-type interfaces (ethX, rfX, tunX) and a set of rules (filters) which allow placing different types of traffic into one or another switching group. Every node can have several switching groups; same interfaces or group of interfaces can be used in several groups simultaneously. Switching groups activated on different nodes of MINT network that have the same switching group identifier represent a **switching zone**. "Switching zone" exists only within MINT network segment.



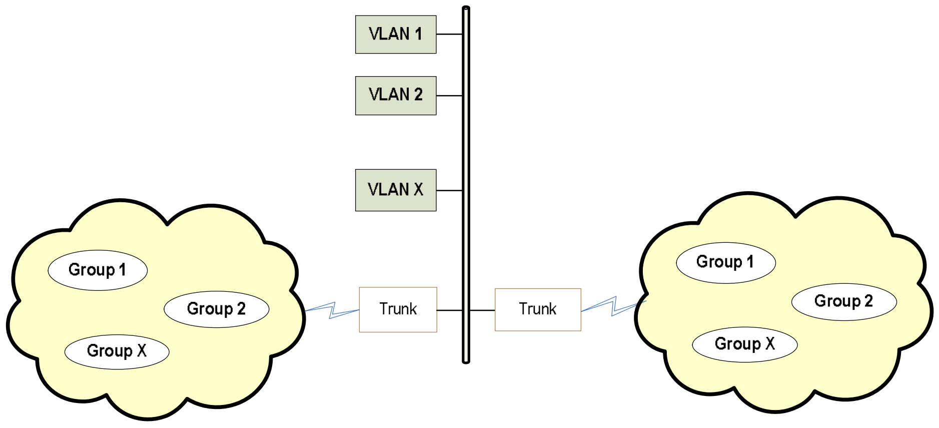
Therefore, MINT network can be viewed as one virtual distributed switch where border nodes act as external ports of the switch. Switch task is to transparently transport packets from one external port to another one (other ones). Important to understand that switching groups should be created only at the nodes where packets come to or from "outside" network ("outside" relative to MINT). On the repeater nodes there is no need to create switching groups regardless the topology.

In order to put an incoming packet to one of the switching groups, a set of flexible rules is available which allows sorting packets according to a various number of criteria (e.g vlan tag, protocol, addresses – MAC/IP, ports and other specific options). **Once put in one of the switching groups packet will never leave it until it reaches one of the external ports**. Having said that, it follows that a set of rules that puts a packet to one of the switching groups is applied only when a packet enters MINT network through one of its external ports. When leaving the network no rules are required as the packet already belongs to one of the switching groups and it will be automatically switched to external port(s) that belongs to the corresponding switching group. Important exception: packets originated by MINT network nodes (e.g. RIP/OSPF or ping) do not belong to any of switching group, therefore they cannot leave MINT network via switching through any of external ports. However, if this is required, on the particular node a special **admin group** can be created in order to send this kind of traffic.

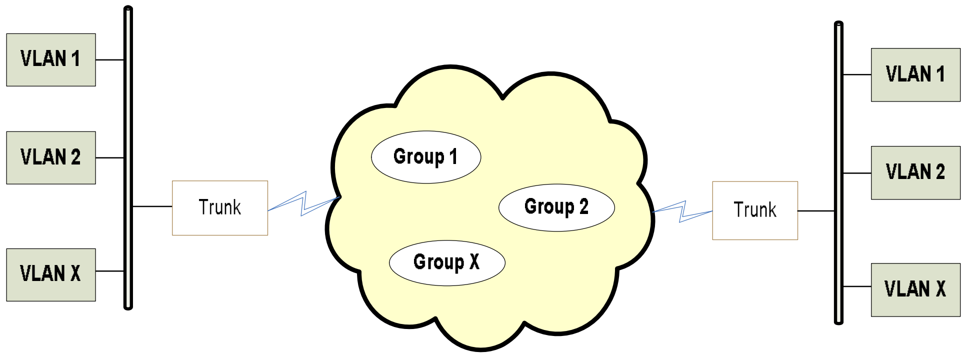
Any group can be marked as a **repeater** group. In this case our switch turns into a simple hub. This mode is effective on point-to-point links.

2. Trunk groups

In order to interconnect several independent MINT networks in one switching segment using wired switches one can use special **trunk** groups. Trunk group is used to send information about switching group number between two independent MINT networks. In this case when sending data to wired switch, a group number is automatically converted into VLAN TAG of 802.1q standard and, visa versa, when receiving a packet from wired interface VLAN TAG is converted to a corresponding group number.



Trunk groups may also be used to solve the task of connecting several VLAN segments.



As packets originated by internal MINT network nodes do not belong to any of the switching groups, special command “**switch local-tag N**” is used to send them through the trunk group.

Also, special rules on interfaces allow flexible manipulations with VLAD ID tags – deleting, assigning and re-assigning.

3. Studying example

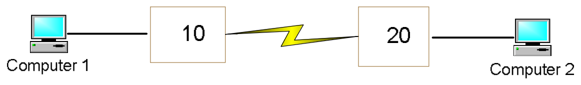
Switch transparently sends Ethernet 802.3 frames (including 802.1q VLAN, broadcast and multicast packets) within the switching group.

Based on MINT convergence sub-layer, the switch optimizes broadcast traffic going through the mesh network and chooses optimal paths for it.

Router works over any network interfaces supporting IPv4 by means of corresponding IP-datagram sending protocol through specific types of network interfaces. With this, broadcast and multicast traffic spreading is limited within a local segment of the interface. This allows separating routed and switched areas of the network.

Each incoming packet is first analyzed by the switch. If the packet matches one of the switching groups it will be correspondingly processed. If not, the packet will be analyzed by the routing module which will decide what to do with a packet. By using these capabilities one can build a hybrid system where a part of the traffic is switched and another part is routed.

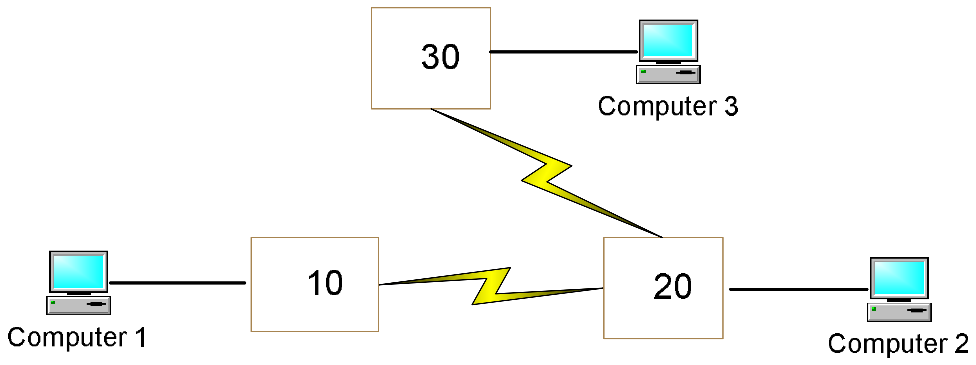
As an example, let's configure a switched point-to-point link.



As we have already configured radio connectivity, let's configure the switch:

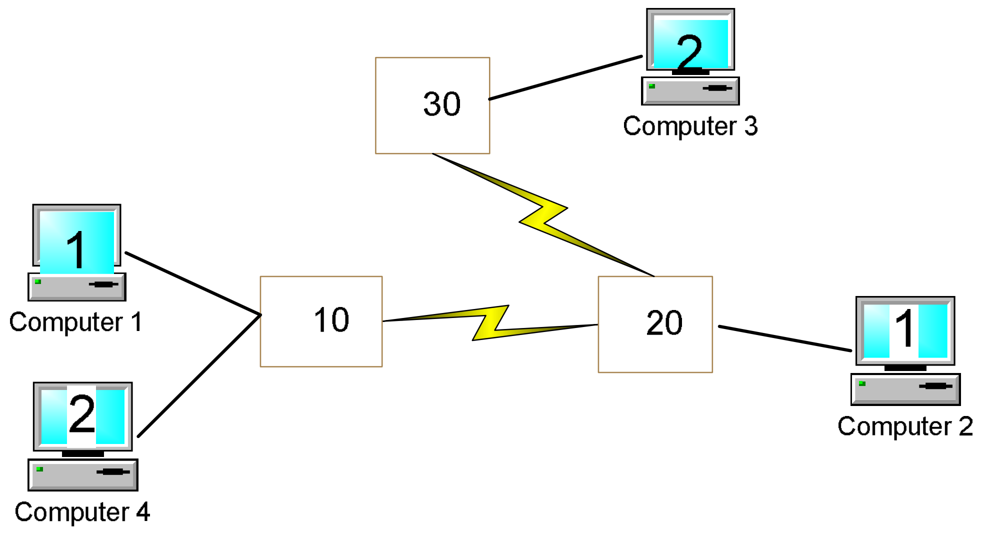
```
sw group 1 add eth0 rf4.0
sw group 1 start
sw start
```

That's all we have to do.
We have created a group 1, added two interfaces, started the group and started the switch. Having configured this set of commands on both units in our point-to-point link, Computer 1 and Computer 2 will be placed into one segment of a network.
This is a trivial case of switch usage when we have two or more interfaces put into a group with no filters and no rules. In this case all packets coming to the unit's from Computer 1 will be delivered to the opposite side of the link to Computer 2. And visa versa.
Let's add another wireless node with the same switch settings.



Regardless of the wireless topology of our network, all three computers will be in the same network segment.

In our next example we want to connect two independent subnets.
We want to connect Computer1 with Computer2, and Computer 3 with Computer 4 but in such a way so that the traffic between these two pairs of hosts was isolated.



Let’s place computers C1 and C2 in a switch group1, C3 and C4 – in switch group 2.

Unit 30 switch configuration:

```
sw group 2 add eth0 rf4.0
sw group 2 start
sw start
```

Unit 20 switch configuration:

```
sw group 1 add eth0 rf4.0
sw group 1 start
sw start
```

On unit 10 we will need two switch groups like this:

```
sw group 1 add eth0 rf4.0
sw group 1 start


sw group 2 add eth0 rf4.0
sw group 2 start


sw start
```

Now we need to divide two streams from Ethernet into two different groups. For this purpose we will use filter rules. But first we need to decide what will be the separation criterion.

Suppose C1 and C2 are in IP-subnet of 10.0.1.0/24; C3 and C4 are in IP-subnet of 10.0.2.0/24.

One can use the following rules:

```
switch list RULE1 match add 'net 10.0.1.0/24'  # filtering rule 1
switch list RULE2 match add 'net 10.0.2.0/24'  # filtering rule 2

switch group 1 add eth0 rf4.0
switch group 1 rule 10 permit match RULE1     # switching only for 1.0/24
switch group 1 deny           # deny rest
switch group 1 start

switch group 2 add eth0 rf4.0
switch group 2 rule 10 permit match RULE2     # switching only for 2.0/24
switch group 2 deny           # deny rest
switch group 2 start

switch start
```

That’s it.
When packet from Computer1 reaches node 10 it will have a source address from 10.0.1.0/24 network, so it will be placed in a switch group 1, then sent to node 20 and then switched to wired Ethernet interface and to Computer 2. Reply packet, after being received by Node 20, will be placed into switch group 1 (as no other options will be available), after that it will be sent to Node 10 and then to Computer 1.

One can use other criteria in order to separate several streams from one another.
For example, you can configure your rules using VLAN tags if they are used in your network. Computer 1 send packets with VLAN tag 100, Computer 4 – with VLAN tag 400. Our rules in this case may look as follows:

```
switch list VL100 numrange add 100      # filtering rules
switch list VL400 numrange add 400      # you can use any names

switch group 1 add eth0:100 rf4.0:0
switch group 1 rule 10 permit vlan VL100   # switching for VLAN 100 only
switch group 1 deny           # deny the rest
switch group 1 start

switch group 2 add eth0:400 rf4.0:0
switch group 2 rule 10 permit vlan VL400   # switching for VLAN 400 only
switch group 2 deny           # deny the rest
switch group 2 start

switch admin-group 1
switch start
```

Please pay attention that when we added interfaces to the groups we put them in **rf4.0:0** and **eth0:100** format. This is to demonstrate VLAN tags manipulation capabilities. This means that when the packet is sent to rf4.0 interface it will be untagged even if tag existed. This can be used, for example, when another side of the network (Computer 2) does not support VLAN tags. Same effect could have been achieved if the following configuration was made on Node 20:

```
switch group 1 add eth0:0 rf4.0
```

Also, the packet coming from MINT network with group number 1 will be sent to wired segment with VLAN ID 100 as requested by our rules. **VLAN tag manipulations are done when packet is sent through a corresponding interface**.

If VLAN tag modifier is not used after interface's name, the packets pass through unmodified. "**:0**" modifier untags all packets, "**:N**" modifier tags the packet with VLAN tag N regardless its previous tag.

Last example can be simplified if we use trunk group:

```
switch list VLANS numrange add 100,400    # create filtering rule


switch group 1 add eth0 rf4.0:0
switch group 1 rule 10 permit vlan VLANS   # switching for VLANS only
switch group 1 deny          # deny the rest
switch group 1 trunk on          # trunk group
switch group 1 start


switch admin-group 1
switch start
```

Node 30:

```
sw group 100 add eth0 rf4.0
sw group 100 start
sw start
```

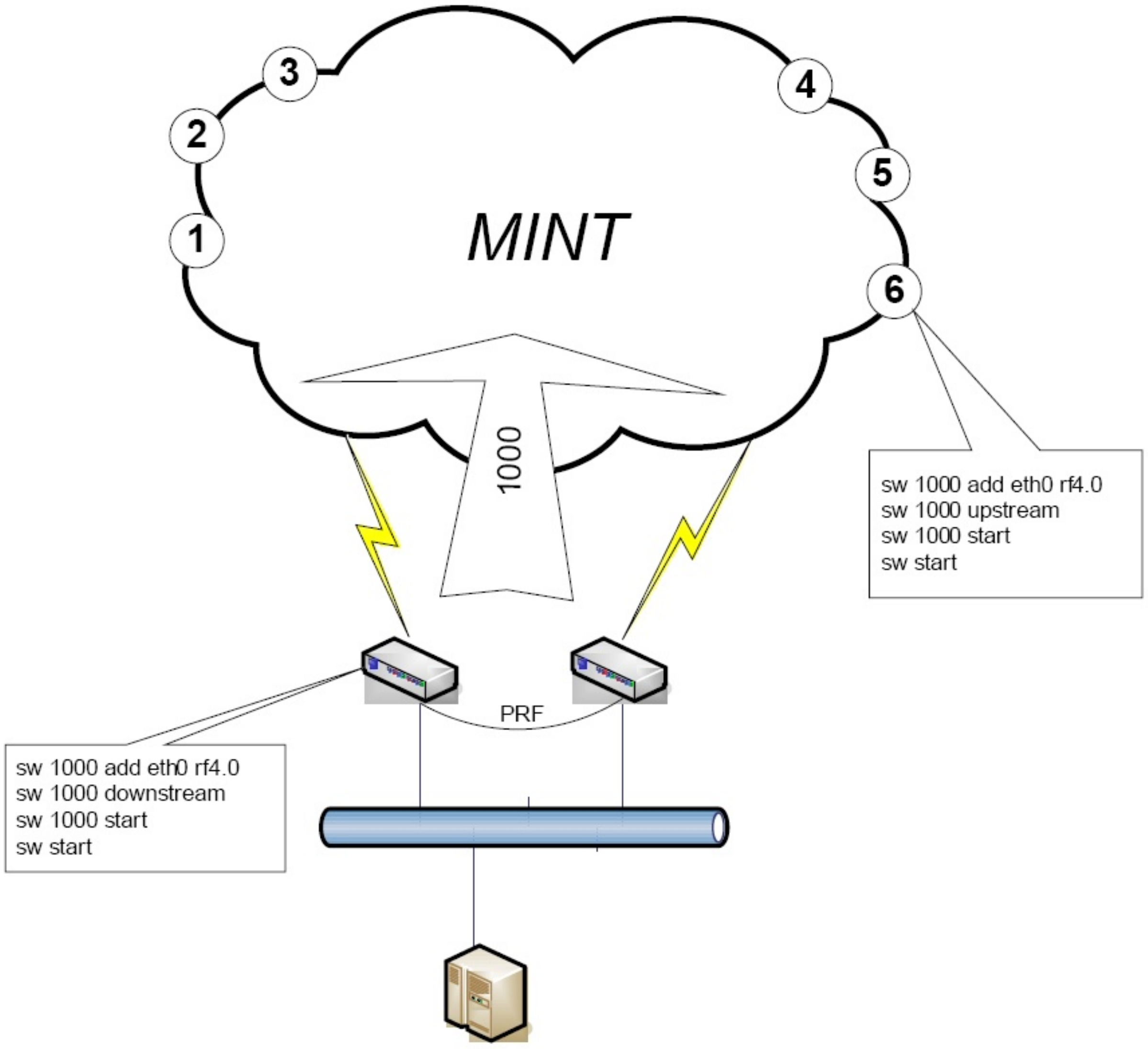
Node 20:

```
sw group 400 add eth0 rf4.0
sw group 400 start
sw start
```

In this case, a packet received through eth0 interface and tagged with VLAN tag N will be automatically placed to switch group N (no need to create this group on this node) and sent to the MINT network. At the same time, a packet coming from MINT network within a switch group N will be sent to wire interface tagged with VLAN tag N. The number of trunk group does not play any role (this will be modified in future releases)

4. Video solution switch configuration.

In video Surveillance systems special attention should be made to transmitting upstream multicast flows. To succeed in this task InfiNet Wireless routers have special switch mode that can be enabled with the following switch configuration using – **downstream** и **upstream** parameters.



```
sw 1000 add eth0 rf4.0
sw 1000 upstream
sw 1000 start
sw start
```

```
sw 1000 add eth0 rf4.0
sw 1000 downstream
sw 1000 start
sw start
```

Imagine video cameras connected to nodes 1,2,3,4,5 и 6. These video cameras transmit video with multicast packets. All these packets need to be transmitted to video server with the most optimal way possible avoiding network flooding.

All downstream (from server to cameras) traffic is transmitted in group 1000. This group contains all network nodes.
Upstream flows from each camera are transmitted directly to nearest to the group concentrator.

Such solution allows to set several concentrators with the same group number. For eliminating broadcast storm that could be possible because of the fact that concentrators are connected to the ports of the same switch – trunk and downstream concentrators never use each other for traffic transmission. Moreover, “upstream” option garantees that end nodes will choose the shortest way to the nearest concentrator. This allows to deploy nodes even on mobile objects. Joining of “downstream” concentrators with the help of pseudo-radio interface (described below) allows maintaining robust connectedness in the whole network.

To show video network groups configuration and their relations use “**mint map swg**” command:

```
=============================================================
Interface rf4.0
```

```
Node 001195F28E78 "NODE1", Id 1250, NetId 0, (master)(polling)
Freq 5320, Sid 10101010, autoBitrate 54000, Noise floor -96

GROUP 1 : sent 1199328 (dynamic) (in-trunk 1000)
   000435FFB7AD "Sklad CPE RF4.0    " 1 hops, Cost 68, (alive 4)

GROUP 2 : sent 2033943 (dynamic) (in-trunk 1000)
   00179AC2F4E2 "Office RF4.0           " 1 hops, Cost 51, (alive 5)

GROUP 5 : sent 18740 (dynamic) (in-trunk 1000)
   000E9B9AEB96 "Garage                  " 1 hops, Cost 51, (alive 4)

GROUP 1000 : sent 347232 (trunk) (in-trunk 1000)
    000435FFB7AD "Sklad CPE RF4.0      " 1 hops, Cost 68, (alive 4)
    000E9B9AEB96 "Garage                    " 1 hops, Cost 51, (alive 4)
    00179AC2F4E2 "Office RF4.0            " 1 hops, Cost 51, (alive 5)
=====================================================
```

This table shows that current node has 5 groups: 1 trunk and 4 dynamically included in trunk zone. It is also shown what nodes are included in each group.

5. Management by means of switching

With switching we can assign address to nodes' wireless interfaces from the same subnet as administrator's network (192.168.0.0/24) and there will be no need to configure routes. But! In this case on all of the network nodes the switch should be configured in such a way so that it has at least one switch group that includes wireless interface (on which IP-address is assigned) even if this interface is the only one in the group:

```
sw g 1 add rf4.0
sw g 1 start
sw start
```

It is necessary so that a switch could educate itself and correctly choose the direction for reply packets to be sent. Group number does not play any role; any of existing groups with corresponding wireless interface can be used.

On the border node to which administrator's network is connected, we will definitely need a switch group which includes both wired and wireless interfaces. And this group's number should be specified in a special parameter of the switch **admin-group**:

```
sw g 1 add eth0 rf4.0
sw g 1 start
sw admin-group 1
sw start
```

Packets originated inside the network can leave this network only by means of admin group. There can be several admin groups on different nodes. Remember that all broadcast packets created by MINT network nodes will go to the wired network through admin group.

A special attention should be paid to the packets destined for the node itself (including broadcast packets a copy of which is also sent to the routing module of the unit). By default these packets are sent to higher layers untagged which allows for the routing module to process them regardless of set VLAN ID as if they were received through ethX interface. If administrator wants these packets to be delivered to a higher level with VLAN ID, the following command should be used:

```
switch self:N
```

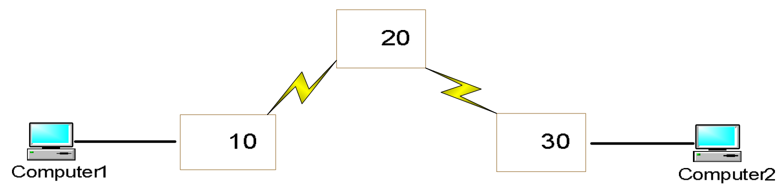
In this case “**self**” acts as a pseudo-interface “looking” inside the unit. Manipulation rules for VLANs are the same as for real interfaces (except for “default”):

```
switch self:0   # untagging (by default).
switch self     # does not change VLAN tag (if it existed)
switch self:N   # sets VLAN tag N
```

Important to remember that packets sent to higher layer (routing module) with VLAN tags can be processed only with properly configured **vlanX** interfaces. For example, if incoming packet has VLAN ID 100 it can be accepted by the system only through vlanX interface configured in the following way:

```
ifconfig vlanX vlan 100 vlandev eth0 up
```

Also note that for repeater nodes (Node 20 in the diagram below) switch configuration is not required – MINT protocol configuration will suffice. Theoretically, this kind of nodes might not have any IP-address.



However, as administrator needs to manage the network IP-addresses most likely will be used (InfiNet Wireless units allow to control all nodes of the network via SNMP even without IP-addresses assigned to units… see below).

How should the access to the network be organized? Two options are available: via routing and via switching. Depending on your scenario you choose the optimal option.

III. Routing in MINT network

Routing way is evident.

As all wireless MINT interfaces are in one virtual Ethernet segment, they can be enumerated by assigning IP-addresses from one IP-subnet (it can be done manually or by means of DHCP). This is good enough already as getting to one node (via telnet, for example) we will give the opportunity to access all other nodes. The access from “outside” can be organized by configuring routing in such a way so that “inner” MINT network can be accessible from administrator’s computer through Ethernet port of the nearest MINT node; from the “inner” MINT network the route to administrator’s computer should go through the radio-interface of the border router.

Imagine that all wireless MINT interfaces of the network have IP-addresses from 10.0.0.0/24 subnet. Administrator’s network is 192.168.0.0/24 and it is connected by Ethernet to one of the nodes which has 10.0.0.1 address assigned to its Ethernet interface. Let’s assign 192.168.0.1/24 address to Ethernet interface of this node. Then, on administrator’s computer (e.g. 192.168.0.2) a route to “inner” network should be configured:

route add 10.0.0.0 netmask 255.255.255.0 192.168.0.1

At the same time, on ALL of network nodes a route to administrator’s network should be enabled:

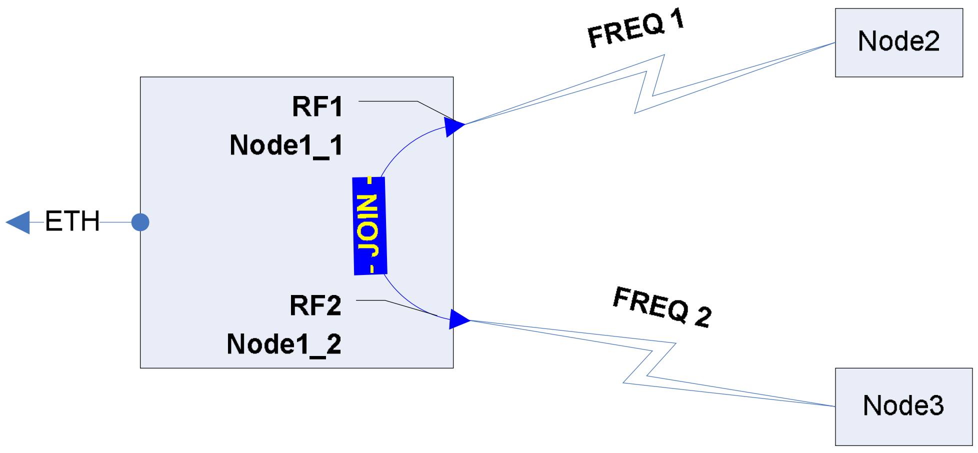
```
route add 192.168.0.0/24 10.0.0.1
```

IV. Additional MINT capabilities

1. Join

Among all of exiting features of MINT architecture one can mention **join** capability which allows joining several interfaces of one unit into one mesh network.

Some unit can have two or more radio-interfaces of different types. Each of these interfaces may act as an independent MINT network node. However, the nodes from different networks will not be able to connect to each other as radio interface parameters will be different (frequencies, modulations or other parameters) and other limitations (authentication parameters, secret keys etc) will take place. **JOIN** function allows for two or more radio-interfaces of one unit to interconnect with each other as if they are two nodes of one network.



mint join rf4.0 rf4.1

```
mint map
============================================================
Interface rf4.0, node 000000000011 "Node1_1" id:11 (mesh)

 2 Neighbors:
------------
  00020 Node2          000000000002, Cost=40 , I/O=24/27 <36/36> /mesh/
  00012 Node1_2         000000000012, Cost=3  , I/O=0/0    <0/0>  /join/


Interface rf4.1, node 000000000012 "Node1_2" id:12 (mesh)

 2 Neighbors:
------------
  00020 Node3          000000000003, Cost=40 , I/O=24/27 <36/36> /mesh/
  00030 Node1_1         000000000011, Cost=3  , I/O=0/0    <0/0>  /join/
```

As you can see from the example, each interface shows two neighbors. In reality the exchange of data between interfaces does not involve their physical activity, energetic parameters of the connection (signal levels and data rates) are not shown (zero). And this kind of connection has a constant and a very low cost.

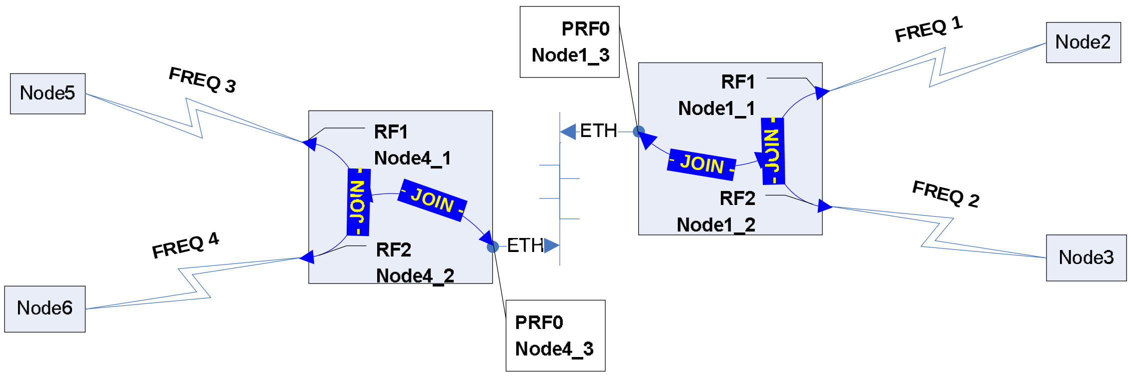
2. Prf (Pseudo radio interface)

More than that, MINT protocol can work not only via radio but via wired Ethernet interface. "**prf**" pseudo radio-interface is used for this purpose. This interface can be "attached" to the physical interface like it's done with vlanX interfaces.

```
prf 0 parent eth0
ifconfig prf0 up
```

Such **pseudo radio-interface** can be configured as a MINT network node and even can be joined with other interfaces. From MINT protocol point of view, this pseudo interface will look like a usual radio interface through which a node can find neighbor and establish a connection with it.

```
mint prf0 start
mint join rf4.0 rf4.1 prf0
```



In this example we have joined several distributed (possibly even geographically distributed) network segments (in previous examples we had a similar situation and the solution was in using trunk groups; **join+prf** combination is much more convenient). Combining "joins" for different radio interfaces and pseudo radio interfaces the network will obtain a good number of alternative paths in any direction thus providing with the best delivery quality and less bottlenecks.

Important. If several interfaces are used with "join" function, when configuring switch groups only one of those joined interfaces should be mentioned in switch group configuration.

```
mint join rf4.0 rf4.1
switch group 1 add eth0 rf4.0
```

Switch works only with independent interfaces. When several interfaces are joined, any of them can be used to "represent" the whole set of interfaces.

V. Topology Management

Here we will discuss how topology can be managed and how streams optimization can be carried out.

First of all, every node of MINT network can be one of three types: **master, mesh** or **slave**.

MASTER:
Master can establish connections with all other types of nodes. He is able to form a network of any topology with other masters or with nodes of **mesh** type.

On master node a marker access (**polling**) can be enabled. Only one master in a network segment can have this option enabled by means of which forming a star-topology segment (point-to-multipoint). With this, all other nodes break their connections with their respective neighbors (with exception of connections formed by **join**)

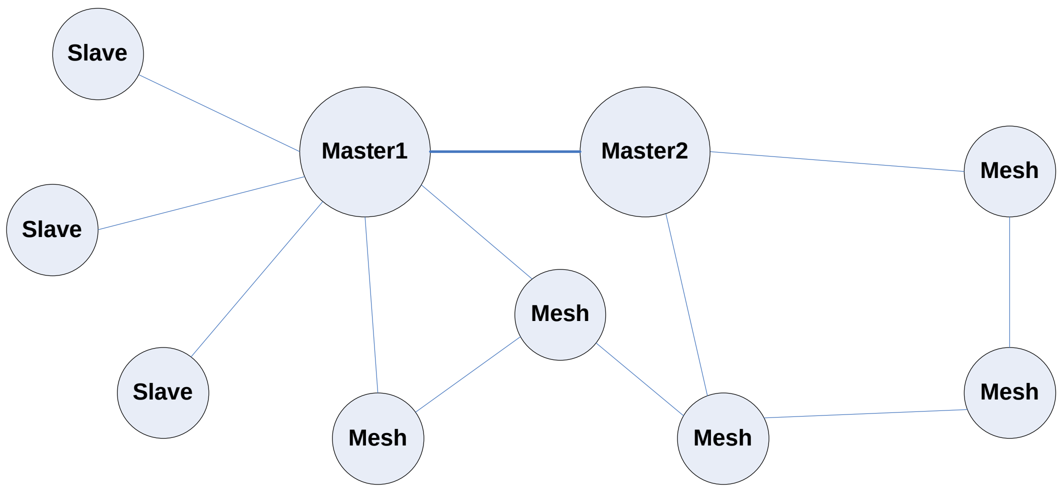
Master type of node is usually used at the core locations of the network in order to process large amounts of traffic. These nodes usually have a fixed geographical location.

MESH:
This type of node can be a part of a network with any topology. Establishes connections with **master** and **mesh** nodes. The difference between master and mesh is that master nodes will try avoid sending traffic of core transport network (master-master) through mesh nodes by setting the cost of master-mesh connection (from master side) higher (**meshextracost** parameter). Thus, mesh type of nodes can be used on mobile units where connection's parameters might change quickly in time.

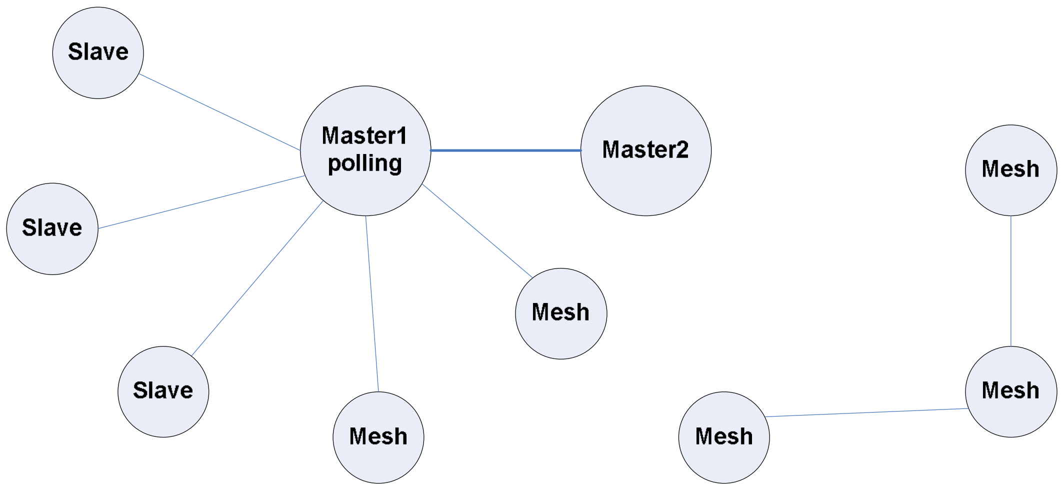
Mesh nodes can work in a marker access node with master node being a polling master. With this, if master turns polling on, mesh node breaks all its connections with other nodes but the master (except join connections). If master node drops off or turns marker access off, mesh node restores its connections with its neighbors (if these connections existed).

SLAVE:
Can only connect to the node with **master** type. With loss of the connection attempts to restore the connection to the master node. Slave node can work with master node using marker access in a classical point-to-multipoint topology.

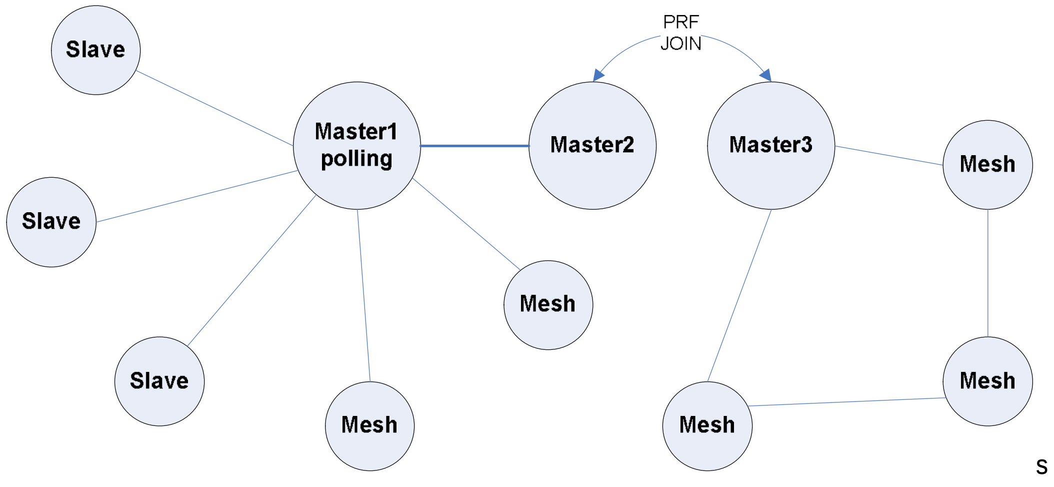
A network with all three types of units may look as follows:



Now, if Master1 turns marker access (polling) on, the network will disintegrate into two separate segments:



Obviously, this is what we need to avoid. We can connect a second segment using "**join**" between two interfaces of the Master (meaning that Master2 and Master3 are two interfaces of one dual-radio unit). Another way is to install a separate unit together with Master2 and connect them by Ethernet and then use **prf** interface to connect two segments into one MINT network.



Apart from setting a node type, one can adjust the cost of connections for the interface or for connections individually. Four different parameters can be configured in order to perform this adjustment:

extracost – is configured for the interface. Sets an extra cost for all connections on this interface. The value of the parameter is added to the cost automatically calculated by MINT protocol. Value of this parameter can only be positive. Zero value disables the parameter.

```
mint rf4.0 –extracost 60
```

meshextracost – is configured for the interface. Sets an extra cost for all connections of master node with its mesh nodes. 500 – by default.

mint rf4.0 –meshextracost 300

joincost – is configured for the interface. Sets the cost of all connections on the interface which were established by means of join (3 – by default). Zero value disables the parameter.

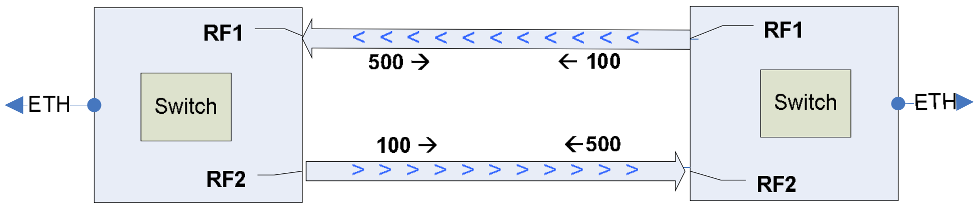
mint rf4.0 –joincost 60

fixedcost – is configured for the interface. Assigns a fixed cost to all connections on this interface (except "join"). Zero value disables the parameter.

mint rf4.0 –fixedcost 500

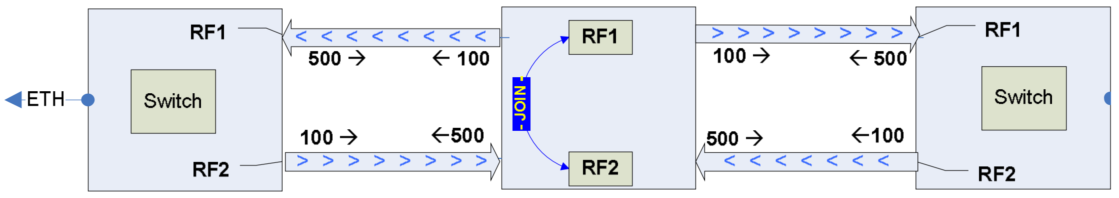
These adjustment parameters are being assigned in the order described above in case of ambiguity.

These parameters can be used to solve specific tasks by creating configurations with pre-defined or preferred flows of data. In the following example we have configured a scheme which imitates a half-duplex channel where opposite streams use separate links.



Thus, a stream from left to right will use a lower link as its cost is less (100) than for the upper one (500). And visa versa for the opposite stream. Switch is aware of cost parameter and takes it into consideration when making a decision.

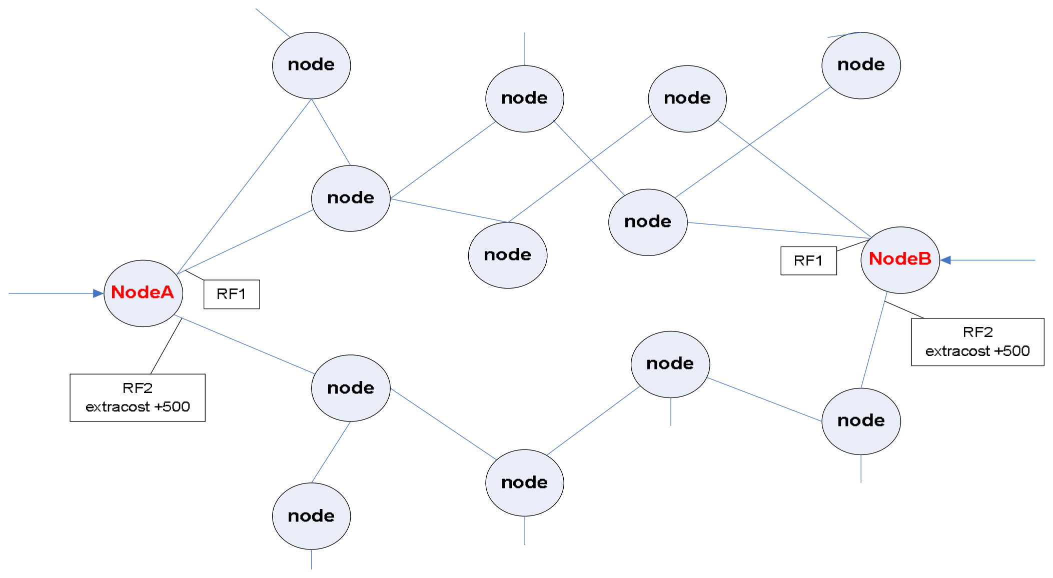
Taking this idea further, a multi-hop link can be created as it is shown on the picture below.



Here we applied "join" function on the middle node so that different interfaces will be chosen for transmitting and receiving.

As it is seen from this example, upper interface of the middle node (repeater node) is always used for transmitting only and, obviously, it does not have any problems with collisions and performance losses. However, lower interface is used for receiving only, so we can get lots of collisions as data is coming from different sources ("**hidden node**" problem due to the fact that left and right nodes in our scheme do not hear each other). The right solution is to turn marker access (polling) on the lower interface. Prior to that the type of the interface should be set to **master**.

Another example of correcting parameters usage is defining preferred paths for the streams:



For example, we want the streams between NodeA and NodeB to use upper side of our network segment so that lower part of the segment would be used for inner streams of the segment. This is accomplished by using "**extracost**" parameter as shown on the diagram (the value of **extracost** parameter should be chosen based on other costs on the node).

VI. Roaming

For smooth and flexible migration between several independent network segments, MINT architecture has **frequency roaming** support. Roaming is disabled by default and the node is working with fixed radio interface parameters.

Any node(s) can be configured as a **roaming leader** (in MINT terminology) which will define radio parameters for the network.

Roaming leader is also working with fixed radio interface parameters, however the information about having roaming leader in the network is being sent to all other nodes. If one segment has several roaming leaders, their radio interface parameters must be identical.

The rest of the nodes can use roaming in order to search for the acceptable roaming leader or the network which has a roaming leader. The search is made by sweeping radio and administrative parameters which are configured in roaming profiles. Each profile is a set of different parameters which are set on each iteration of the search.

Heuristic search algorithm quickly evaluates the situation in every network it finds and, eventually, selects one network which suits best. The aim of the search is not finding a roaming leader but a network which has roaming leader.

If roaming leader parameters change or all connections to the network are lost, the search will be started over.

Moreover, roaming leader supports **DFS** and **Radar Detection** (if permitted by the unit license).

Key profile parameters:

freq XXX – radio interface frequency. "auto" keyword can be specified – in this case all supported frequencies will be used.

sid XXX – SID of the network.

bitr XXX – bitrate of radio interface. Acts as an upper limit for the bitrate if autobitrate mechanism is enabled.

bandwith {full|half|quarter} – channel width of radio interface

key XXX – key of the network

```
mint rf4.0 profile 1 -freq 5920,5960 -sid ABCDE –key mykey
mint rf4.0 profile 2 -freq auto -sid DEAD –key secret
mint rf4.0 roaming enable
```

However, for the service providers it is more important to have IP roaming not frequency roaming. This would mean providing services regardless of the client’s connection point to the network. This task is best solved at IP routing level. In order to provide with more flexibility in solving this task, MINT technology offers several capabilities.

Many components of the system are integrated in MINT architecture and are able to use it. Here comes a description of one of the possible scenarios.

A node with enabled roaming is scanning the media and connects to the selected network thus becoming a MINT node (Ethernet host).

DHCP client, having receiving the information about network connectivity, starts searching for the server in order to receive the configuration. DHCP server might be organized either on one of MINT network nodes or outside (through **admin-group**).

After radio interface is assigned IP-address (manually or via DHCP) OSPF protocol comes into picture. Using ”**autointerface**” feature, OSPF detects a new address in the system and immediately creates a logical interface and starts looking for new neighbors thus integrating the node into existing IP infrastructure.

In order to increase network utilization effectiveness, a new interface type “mesh” was created for OSPF protocol, which “sees” all nodes connections to its neighbors as point-to-multipoint. Even though a network is a plain Ethernet segment, “**mesh**” interface of OSPF permits having different costs to its neighbors using MINT networking layer information. So, OSPF costs automatically become optimal! Moreover, using “**mesh**” OSPF interface decreases processing and calculation overheads.

Also, “**mesh**” OSPF interface can be used to avoid a “bottleneck” problem when traffic goes “outside” through one single link. Now, network can have multiple default gateways; with this every node (OSPF router) will chose the most optimal gateway.

How difficult is to configure the unit in the way described? Here follows an example of the configuration (key parameters):

```
ifconfig eth0 1.2.3.4/24 up   # Client LAN network


#MINT configuration
   mint rf4.0 -type mesh
   mint rf4.0 –name Client-NNN
   mint rf4.0 –nodeid NNN
   mint rf4.0 prof 1 -freq auto -sid 10101010 -bitr 36000 –key
mykey
```

```
   mint rf4.0 roaming enable
   mint rf4.0 start

#DHCP configuration
   dhcpc -k test
   dhcpc rf4.0 start

#OSPF configuration
   ospf start
   ospf interface eth0
   ospf interface rf4.0
   ospf  network mesh
   ospf router
   ospf  redistribute-connected
   ospf  auto-interface rf4.0 area 0.0.0.1
   ospf end
```