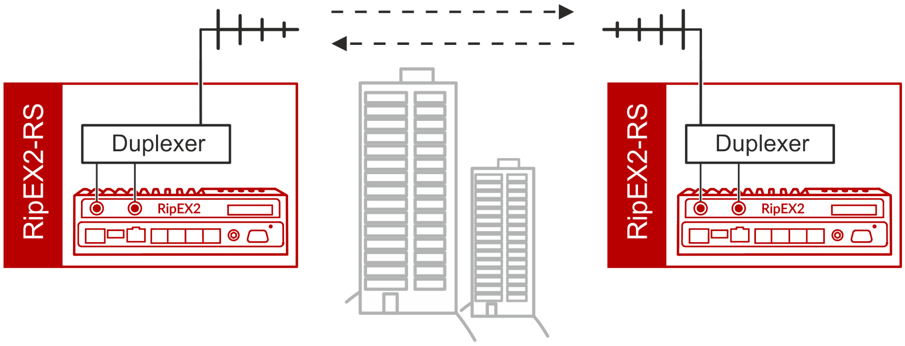
RADIO DATA NETWORKS

Application notes



RipEX2
Babel dynamic routing protocol

version 1.1
2024-08-28
fw 2.1.7.0

RACOM s.r.o. | Mirova 1283 | 592 31 Nove Mesto na Morave | Czech Republic
Tel.: +420 722 937 522 | E-mail: racom@racom.eu

www.racom.eu

Table of Contents

Babel dynamic routing protocol

This application note will guide you through several **Babel examples** and its typical usage. Examples are based on four RipEX2 units only – feel free to accommodate scenarios to your RipEX2 network.

The application note explains various Babel parameters, but it does not consist all of them. Check the RipEX2 manual or help for further information. On the other hand, this application note can provide a valuable insight into important details or more practical explanation than strict information from the manual.

Babel description

Babel is a dynamic routing protocol working on a link layer. Implementation in RipEX2 (and bird daemon) uses **UDP/IPv6 multicast packets on port 6696**.

Babel is an Interior Gateway Protocol (**IGP**) and works within one autonomous system. Routers with Babel enabled must be directly adjacent to each other (on the same link). The protocol can be configured on multiple interfaces (ETH, radio, ...) of each RipEX2 unit and it seeks for connections on every one of them.

Babel works on both, **wired** and **wireless** connections. In wireless networks, it uses efficient multicast transmission, does not require all participants to “hear” each other and uses a finer link quality evaluation instead of binary decision (good/bad).

Individual protocol instances within the network send and receive Hello packets which map interconnections and their qualities. They also exchange routing update messages regularly.

The protocol tries to find the most convenient path according to the metric and not creating loops. Bellman-Ford algorithm is being used. **The metric** is set on the interface and represents a price for receiving packets. This is an abstract value and can be set to any value (e.g., it can be derived from the particular link speed – Ethernet, LTE, Radio link, …). The lower the value, the more advantageous the link/interface is. The maximum value of the metric is 65535 (infinity) and must be greater than 0.

Interface types:

- **Wired** – assumes a reliable link. The quality is evaluated according to the number of received Hello packets. If more packets are lost than the configured limit, the line is considered down.
- **Wireless** – assumes a variable connection quality. A quality is therefore evaluated by a variant of the ETX (Expected Transmission Count) metric. If Hello packets are lost, the price of the interface gradually increases until the line is declared down.

Bird implementation of Babel does not support any security.

If the protocol is used in the Hot Standby (HS), keep in mind it is completely turned off in the passive unit. Once the passive unit becomes the active one, it must first find out all the routes from the start which may take a while (based on settings). Babel can use the virtual shared HS IP address.

Selected Babel parameters

Common – Router offering

- Default: “On”
- It turns on/off propagation of routing rules obtained from its own neighbor. If it is turned off, the station behaves as end terminal (Babel paths start and end here, they are not forwarded through).

Network – Interface

- You must manually set the interface name for each interface you want Babel to be active on. A list of possible interfaces is explained in manual, or see below.
  - LAN – “if_” + interface name defined in GUI (e.g., “if_bridge”)
  - VLAN – “if_” + interface name defined in GUI + ‘.‘ dot and VLAN number (e.g., “if_bridge.29”)
  - Radio – “radio”
  - Hot Standby – “hstdby”
  - GRE L3 – “gre_tunX” where ‘X’ is the tunnel number, starting from zero
  - Cellular – “aux”

Network – Hello limit

- Used only for “Wired” interface.
- A limit of received Hello packets for which the link is considered down (from expected 16).

Network – Advertised next hop

- It is used if multiple IP addresses are set on one interface. Otherwise, use default 0.0.0.0.
- Select which IP address should be used as a “next-hop” IP for our neighbor routing tables.

Brief introduction to examples

In general, Babel is easy to setup. Once you have your RipEX2 units configured by themselves (Radio, ETH IP addresses, Radio protocols, SCADA setup, …), you just need to:

- Enable Babel and set its unique Router ID
- Choose RipEX2 interfaces on which Babel should operate
- Select Babel timing (speed of routes propagation, …)
- Local LANs to be propagated

If you only have two or three RipEX2 units, you can easily accommodate most of the examples as well. You can configure just part of the example, the RipEX2 configuration is mainly the same.

Example 1 The first example depicts a network only with RF channel usage and shows “only” dynamically obtained direct (one radio hop) LAN routing.

Example 2 The second example shows and explains Radio filters to optimize Babel dynamic routing based on current RF channel conditions and Relay filters to optimize how and where are the routes propagated.

Example 3 The third example changes the RF channel setup to simulate using two repeaters for end-to-end communication. The network is not a mesh compared to the first example.

Example 4 The fourth example utilizes Ethernet connection as a quick backbone and radio channel only as a backup option. Feel free to utilize just two RipEX2 units – primary connection is via Ethernet; backup is via the radio channel.

Example 5 The fifth example changes the above Ethernet by LTE and GRE connection.

Example 6 The sixth example shows a network combining Babel and OSPF. OSPF is configured in RipEX_A and RipEX_D units, whereas Babel is configured in RipEX_A, B and C. These two protocols are neighboring in one Autonomous System Border Router “ASBR” (RipEX_A) which divides the whole network into two parts. Bandwidth optimized Babel on the Radio segment and standardized, widely-used and fast OSPF on the Ethernet segment.

Example 7 The seventh example extends the sixth one to show even more from OSPF and Babel combination. There are two ASBR routers between Babel and OSPF network segments. By default, the primary path should go over RipEX_A and a backup path is over RipEX_C. Dynamic protocols exchange routing rules and preferences between each other.

1. Mesh topology

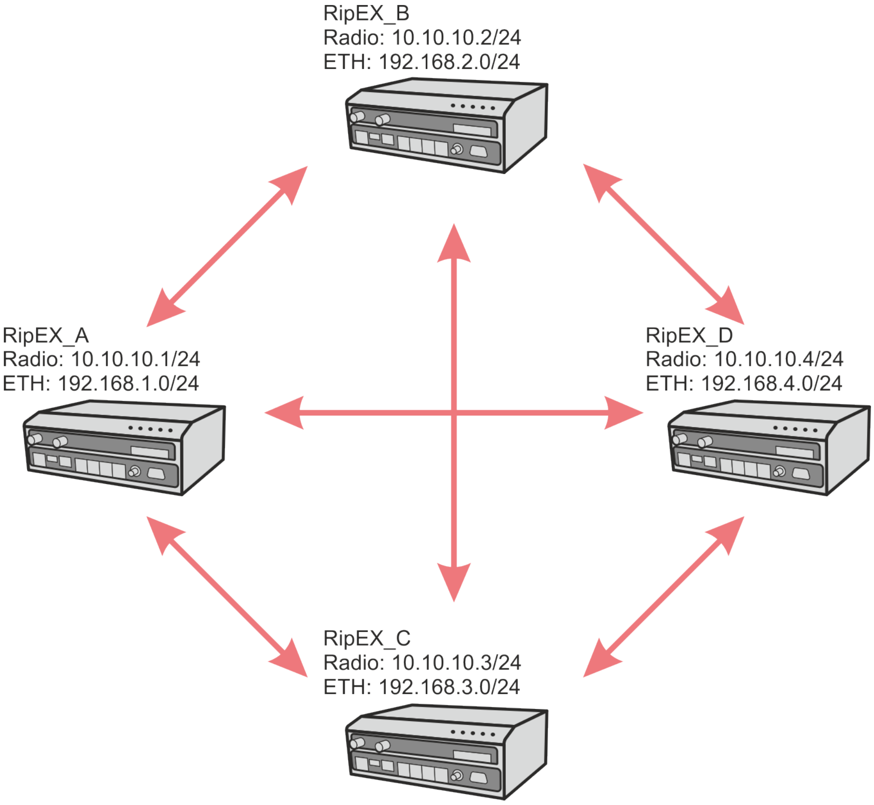


Fig. 1.1: Example 1 – Mesh topology

1.1. Description

The first example shows how Babel can be used to propagate RipEX2's LANs automatically instead of static routes. The topology allows every device to directly see each other resulting in a situation that all communication goes directly between two particular RipEX2 units. There is no dedicated repeater or another technology such as LTE or Fibre optics.

1.2. RipEX_A configuration

Start with unit's Name and Mode. Go to the SETTINGS – Device – Unit menu. Select the "Router" mode, because it is not possible to configure dynamic routing protocols in the Bridge mode. Set the name to "RipEX_A".

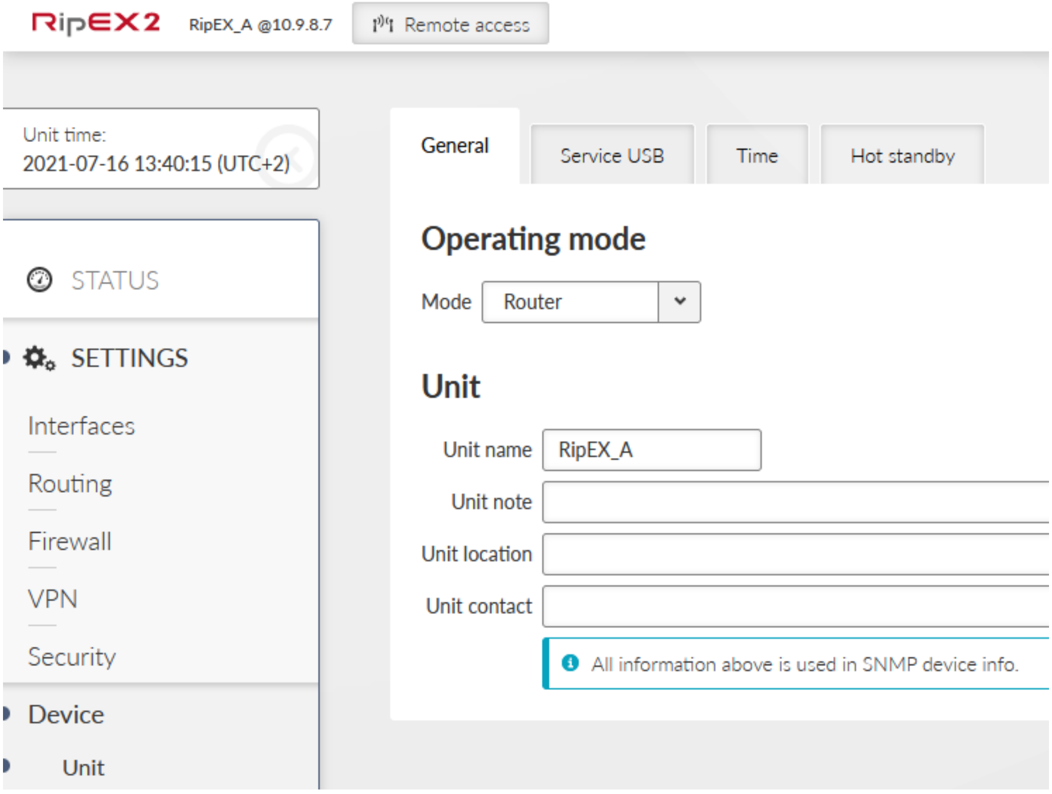


Fig. 1.2: RipEX_A – Mode and name

You can also set a correct time in the unit, either via a working NTP server or just manually update it by the time in your browser (“Update in device” button and a check-box “Use browser time”). This is useful in general and mainly for debugging purposes (Statistics, monitoring, …). It is not required by Babel protocol.

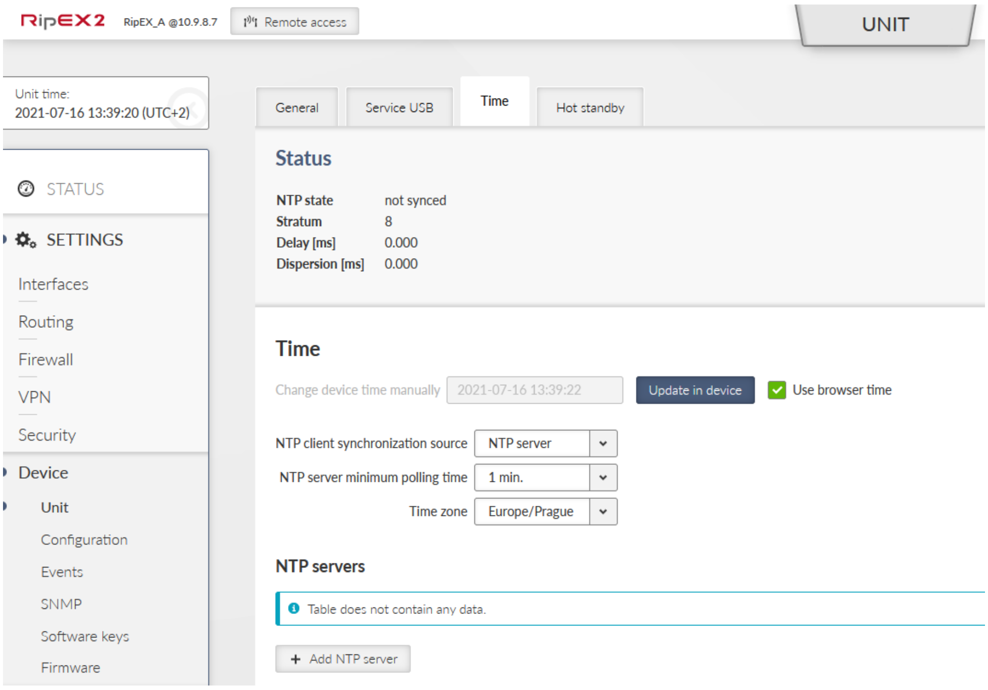


Fig. 1.3: RipEX_A – Time

Configure a correct Ethernet IP address 192.168.1.1/24 (bridged on all ETH ports).

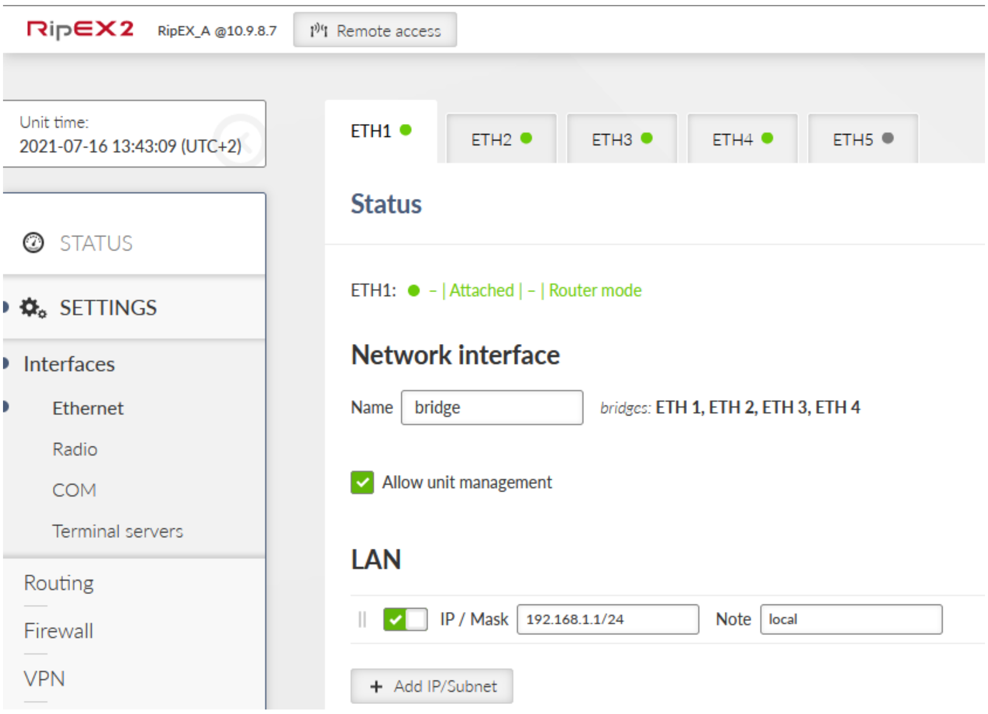


Fig. 1.4: RipEX_A – Ethernet IP address

Go to the Radio submenu and configure the Radio interface. You can accommodate most of the parameters to suit your needs, but be consistent throughout this application note and all its examples.

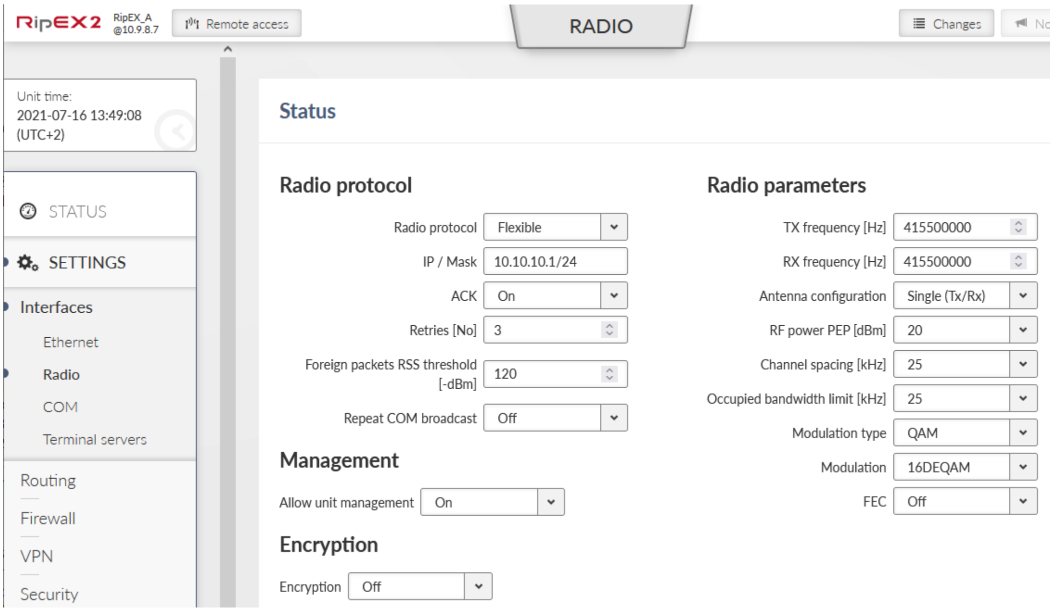


Fig. 1.5: RipEX_A – Radio interface settings

Settings:

Radio protocol	Flexible
IP / Mask	10.10.10.1/24
Radio protocol	Flexible
ACK	On
Retries	3
TX/RX frequencies	415.500.000 MHz
Antenna configuration	Single (Tx/Rx)
RF power PEP	20 dBm (testing on the desk, lowest power)
Channel spacing and OBW	25 kHz
Modulation and its type	QAM, 16DEQAM
FEC	Off

Note

Using the Flexible Radio protocol via a simplex channel (half-duplex operation) might result in a FULL MESH organized network. When BDP Radio protocol is set, using the Babel dynamic routing results in a self-configuring network with a STAR topology. There is no Encryption set, neither any Individual link options within these examples.

The most important menu is SETTINGS – Routing – Babel. Activate the protocol and set the Router ID to unique 1.1.1.1 and enable Routing offering so that it forwards received routes to other neighbor.

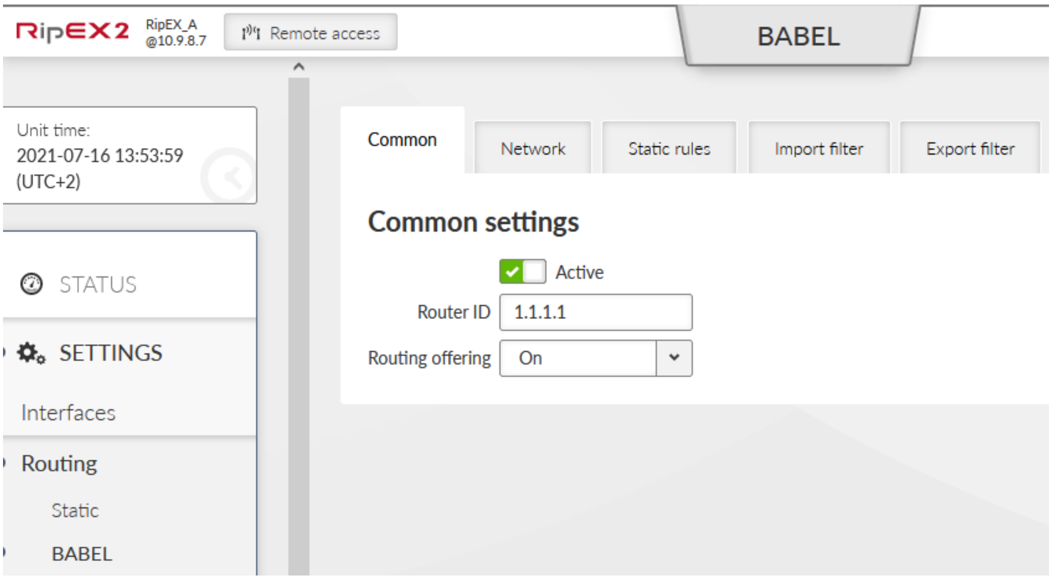


Fig. 1.6: RipEX_A – Babel common settings

Go to the Network panel and create a new Wireless network.

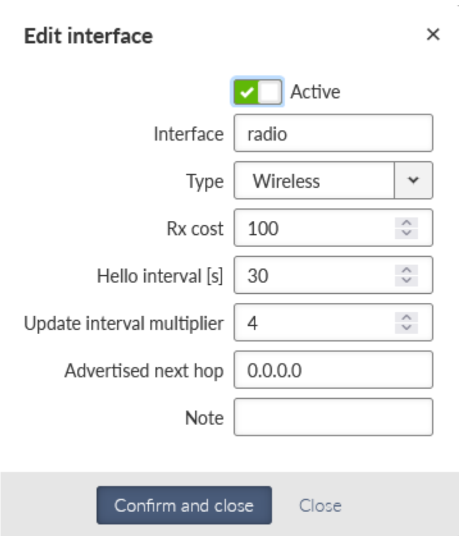


Fig. 1.7: RipEX_A – Babel Network settings

Interface must be “radio” (interface name) and its type “Wireless” (it is a radio channel). Change the Rx cost from default 128 to 100.

Increase the Hello interval to 30 seconds. This is important so that Babel does not send too many overhead packets on the radio channel. Decision on this (and other) parameters is always a “tradeoff”. Lower the interval, quicker protocol operations such as topology changes detection or complete routing propagation, but in a cost of higher Radio channel utilization.

Keep in mind that if you run low FSK or even low QAM modulation, the primary goal is still correct SCADA operation and not overhead data.

Another parameter to be set is Update interval multiplier and is set to default 4. This number multiplies the Hello interval (in our example it is 4×30 seconds = 2 minutes). Router updates packets are sent in ~ 2 minutes intervals.

This setup, the same in all 4 RipEX2 units, results in approximately 1 Babel packet per 10 second, i.e., 100 bps. The topology change can be detected and spread across a network within quite a long interval, ranging from 30 seconds to several minutes. Optimize these values to suit your topology, used modulation type and SCADA traffic.

Go to another panel “Static rules” and configure advertised network 192.168.1.0/24 with default metric equal to 0.

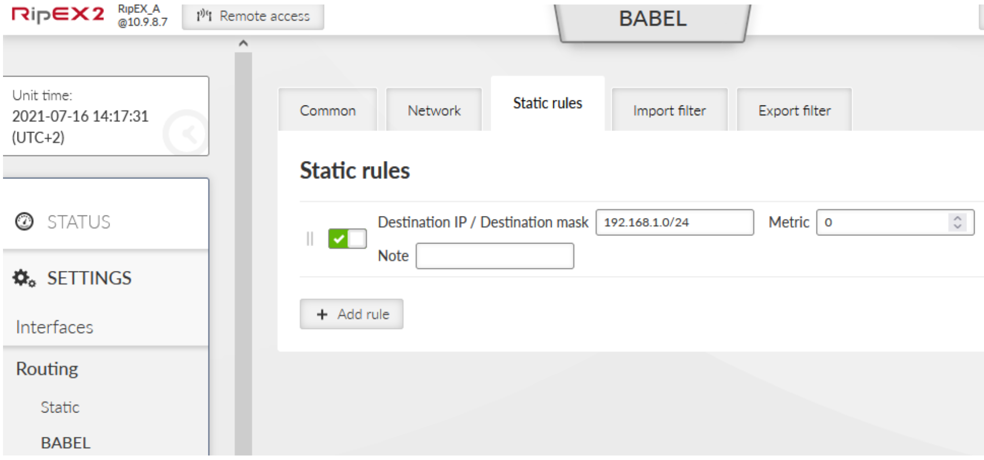


Fig. 1.8: RipEX_A – Static rules settings

Go to another panel “Import filter” and create a new rule. The only non-default parameter we need to set is “Local preferred source address” (LPSA) – set it to 192.168.1.1 (Ethernet IP of RipEX_A). This is a preferred source IP address for locally generated packets.

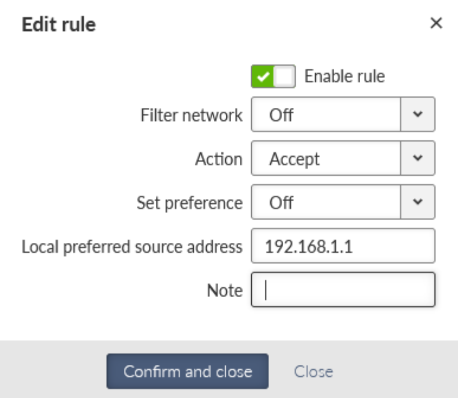


Fig. 1.9: RipEX_A – Import filter settings

We do not configure anything in the “Export filter” panel.

The configuration is complete. Go to the “Changes” menu (upper right corner button) and save the changes.

1.3. RipEX_B, RipEX_C and RipEX_D configuration

Other units share most of the settings we configured in RipEX_A. We just highlight the differences compared to RipEX_A setup.

RipEX_B

Unit name	RipEX_B	
Ethernet IP	192.168.2.1/24	
Radio IP	10.10.10.2	
Babel routing		
	Router ID	2.2.2.2
	Static rules	192.168.2.0/24
	Import filter LPSA	192.168.2.1

RipEX_C

Unit name	RipEX_C	
Ethernet IP	192.168.3.1/24	
Radio IP	10.10.10.3	
Babel routing		
	Router ID	3.3.3.3
	Static rules	192.168.3.0/24
	Import filter LPSA	192.168.3.1

RipEX_D

Unit name	RipEX_D	
Ethernet IP	192.168.4.1/24	
Radio IP	10.10.10.4	
Babel routing		
	Router ID	4.4.4.4
	Static rules	192.168.4.0/24
	Import filter LPSA	192.168.4.1

1.4. Diagnostics and Testing

Once configured, you can wait until the network converges to the correct routing, or you can also reboot all the units so the protocol starts from scratch.

1.4.1. Routing

Go to the DIAGNOSTICS – Routing menu and check the Babel panel. Let's go through individual tables.

BABEL routing

Interfaces

```
BIRD 2.0.7 ready.
babel1:
Interface  State  RX cost   Nbrs   Timer Next hop (v4)    Next hop (v6)
radio      Up         100      3   0.721 10.10.10.1       fe80::202:a9ff:fe20:6f9
```

Fig. 1.10: Babel diagnostics – Interfaces

In the 1 st table, we can see all interfaces on which Babel is either configured or found. If the interface is missing, but should be there, it probably failed while initialization (e.g., because of missing link IPv6 address).

State Interface state, either "up" or "down"

Rx cost It displays configured received cost (for other neighbor)

Nbrs Number of detected neighbors on particular interface

Time Number of seconds until next Babel Hello or Update transmission

Nexthop What "next-hop" address the router offers to neighbors (IPv4 and IPv6)

- Can be useful in case of configured 0.0.0.0 – to check if correct IP is really used

Neighbors

```
BIRD 2.0.7 ready.
babel1:
IP address                 Interface  Metric Routes Hellos Expires
fe80::202:a9ff:fe20:ae3    radio         100      3     16  44.662
fe80::202:a9ff:fe20:531    radio         100      3     16  27.512
fe80::202:a9ff:fe20:789    radio         100      3     16  36.432
```

Fig. 1.11: Babel diagnostics – Neighbors

IP address Neighbor's IPv6 link address

Interface Interface on which the neighbor was found

Metric Current metric for receiving from neighbor

Routes Number of routes received from neighbor

Hellos Number of received Hello packets (from up to 16)

Expires [s] Time until reception of another expected Hello packet from neighbor

Routes

```
BIRD 2.0.7 ready.
babel1:
Prefix                    Nexthop                   Interface Metric F Seqno Expires
192.168.2.0/24            10.10.10.3                radio        200       2 303.655
192.168.2.0/24            10.10.10.4                radio        200       2 339.506
192.168.2.0/24            10.10.10.2                radio        100 *     2 345.424
192.168.3.0/24            10.10.10.3                radio        100 *     2 303.655
192.168.3.0/24            10.10.10.4                radio        200       2 339.506
192.168.3.0/24            10.10.10.2                radio        200       2 345.424
192.168.4.0/24            10.10.10.3                radio        200       2 303.655
192.168.4.0/24            10.10.10.4                radio        100 *     2 339.506
192.168.4.0/24            10.10.10.2                radio        200       2 345.424
```

Fig. 1.12: Babel diagnostics – Routes

A list of all routes from all neighbors. It may even consist “loops” for routes which are not currently used, but once the protocol switches to them, loops are solved and “fixed”. Or in other words, such routes are are not considered as "candidates".

Prefix Range of destination IP addresses for a particular rule

Nexthop Next-hop address

Metric Current cost to the destination

F * for currently active rule, + for feasible next candidate for active rule

Seqno Update packet sequence number which announced the rule

Expires [s] Time until the rule expires

Entries

```
BIRD 2.0.7 ready.
babel1:
Prefix                    Router ID                 Metric Seqno  Routes Sources
192.168.1.0/24            00:00:00:00:01:01:01:01      0      1       0       0
192.168.2.0/24            00:00:00:00:02:02:02:02    100      2       3       1
192.168.3.0/24            00:00:00:00:03:03:03:03    100      2       3       1
192.168.4.0/24            00:00:00:00:04:04:04:04    100      2       3       1
```

Fig. 1.13: Babel diagnostics – Entries

Prefix Prefix of routed address range

Router ID Router ID – first 4 Bytes either zeros, or randomized. Second 4 Bytes equal to Router ID set in the configuration

Metric Current cost to the destination

Seqno Number of various routing rules for particular prefix/subnet

Routes Time until the rule expires

Sources Number of various routers which export a particular prefix to Babel network

Table babel_ipv4

```
BIRD 2.0.7 ready.
Table babel_ipv4:
192.168.1.0/24       unreachable [static_babel 11:33:35.727] * (1000)
        Type: static univ
        Babel.metric: 0
192.168.2.0/24       unicast [babel1 11:50:05.172 from fe80::202:a9ff:fe20:789] (210/100) [00:00:00:00:02:02:02:02]
        via 10.10.10.2 on radio
        Type: Babel univ
        Babel.metric: 100
        Babel.router_id: 00:00:00:00:02:02:02:02
192.168.3.0/24       unicast [babel1 11:48:44.385 from fe80::202:a9ff:fe20:ae3] (210/100) [00:00:00:00:03:03:03:03]
        via 10.10.10.3 on radio
        Type: Babel univ
        Babel.metric: 100
        Babel.router_id: 00:00:00:00:03:03:03:03
192.168.4.0/24       unicast [babel1 11:49:56.247 from fe80::202:a9ff:fe20:531] (210/100) [00:00:00:00:04:04:04:04]
        via 10.10.10.4 on radio
        Type: Babel univ
        Babel.metric: 100
        Babel.router_id: 00:00:00:00:04:04:04:04
```

Fig. 1.14: Babel diagnostics – Table babel_ipv4

The table consist of advanced and detailed information of all routes from Babel table.

1.4.2. Tools

Now, check the remote unit's IP accessibility. Go to the DIAGNOSTICS – Tools – ICMP ping menu.

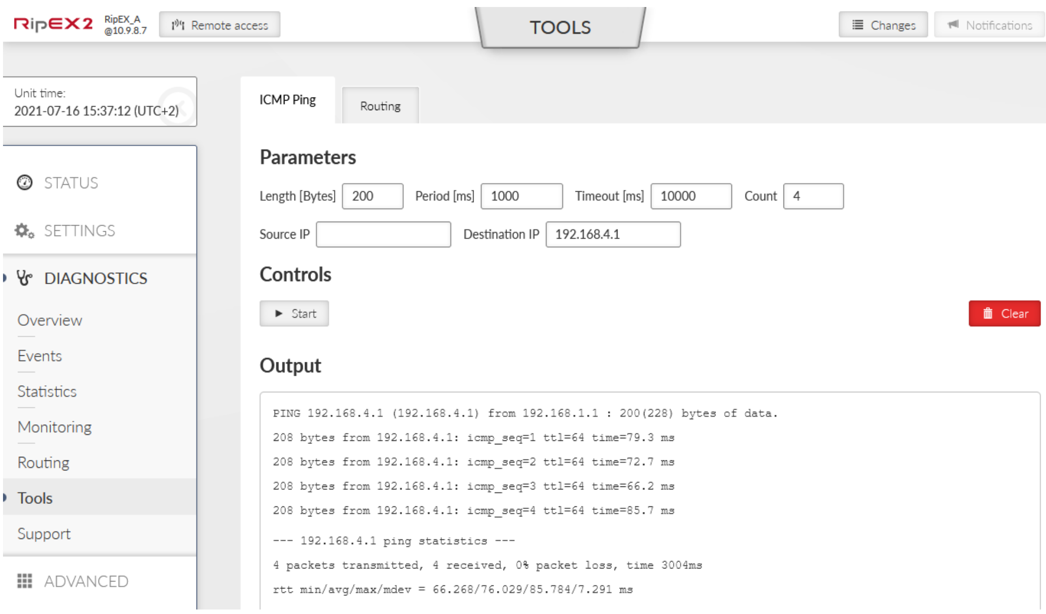


Fig. 1.15: RipEX_A – Diagnostics – Tools – ICMP Ping

Fill in the Destination IP field with required IP (try all remote LAN IPs). Check if it gets through. If not, check the Routing diagnostics for available routes.

You can also check what route is used for particular destination in another panel “Routing”.

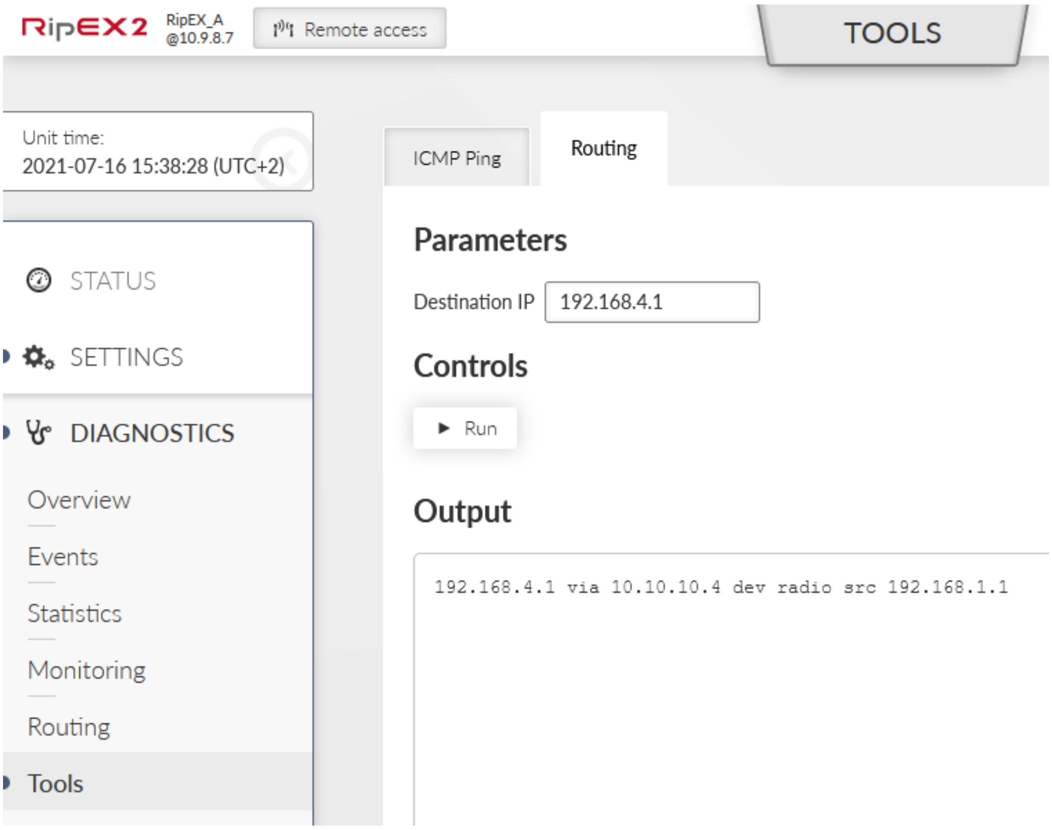


Fig. 1.16: RipEX_A – Diagnostics – Tools – Routing

You can debug issues further in Statistics and/or Monitoring menus.

2. Mesh topology with Radio and Relay filters

This second example uses the same topology as described within the first example.

Since 2.1.7.0 firmware, Babel can control the path priorities based on current RSS/MSE values (radio filter). This is very handy in case the link qualities differ and some of the links are barely usable. With such filter, we can configure RipEX2 units to ignore such bad quality links.

A second improvement is the Relay filter which selects which of the obtained rules are being forwarded and which are not – i.e. if the unit is a repeater/forward node, or the end station (terminal). It can also be used to alter routing rule metrics and thus, prioritize or discriminate some of the nodes/units.

Within the following example, we will configure both the Radio and Relay filters.

Keep in mind all Babel parameters and its filters are well explained within the manual.

2.1. Radio filters

Configure Radio filters in all your networks utilizing Babel protocol via the Radio channel. Why?Neither Babel nor other dynamic routing protocols can take RSS/MSE values into account. In most of the networks, some links have very good RSS and MSE values running high QAM modulations. Links with (very) bad RSS/MSE values should not be used for routing user traffic in case there are links with higher quality. Without the Radio filter, short Babel management/overhead data may be able to successfully go through such links, but important user traffic might be corrupted resulting in unreliable SCADA operation.

Babel itself has a known mechanism of Hello packets and increasing link metrics based on Hello packets success rate of going through particular links. We enhanced this behaviour (proprietary) so that some of the Hello packets are discarded if received with RSS or MSE values below set thresholds.

Default thresholds are:

- RSS
  - Soft: -110 dBm
  - Hard: -130 dBm
- MSE
  - Soft: -10 dB
  - Hard: -5 dB

If the RSS/MSE values of received Hello packet is worse than the “Hard” limit, it is always discarded. In case the RSS/MSE values are better than “Soft” thresholds, they are always accepted.

If the values are between thresholds, Hello packets are discarded randomly with probability increasing linearly between Soft and Hard limits. The probability of a Hello packet being received by the filter is calculated as the product of the probabilities based on RSS and MSE. If at least one of the variables exceeds the Hard limit, the packet is always discarded. The limit setting is either global or individual for each Hello packet source address (radio IP address). The individual setting is only used if there is a translation of the link address to a radio IP address (ARP request/reply for the given IP had to be sent/received so that you don’t see 0.0.0.0 Radio IP within the Statistics menu, but a correct IP for a given Link address). Discarding a Hello packet will cause the Babel protocol metric to be increased and the link to be disadvantaged.

Keep in mind each modulation has its own sensitivity levels. If you combine multiple modulations within the network, individual link thresholds are recommended.

Sensitivity levels (RSS) can be read from *the Specification*[1].

MSE recommendations can be read within *the application note on RACOM website*[2].

Let's see one example. In the whole network, we run the QAM modulation π/4-DQPSK, within the 25 kHz channel. The sensitivity level (for BER 10^{-6}) is -111 dBm. Recommended MSE is -14 dB.

Default thresholds could be:

- RSS
  - Soft: -91 dBm (20 dBm Fade margin)
  - Hard: -111 dBm (sensitivity level from the Specification)
- MSE
  - Soft: -14 dB (i.e. real value at least equal to the recommended -14 dB)
  - Hard: -10 dB (based on user experience, no precise calculation for this value)

Received Hello packets' RSS/MSE values and its operation:

- RSS = -80 dBm, MSE = -20 dB
  - Packet is passed to Babel successfully
- RSS = -99 dBm, MSE = -15 dB
  - RSS value is between the Hard and Soft thresholds and = passing the Hello packet probability = 60 %
  - MSE is over the Soft limit = passing the Hello packet probability = 100 %
  - Resulting probability is 0.6 * 1 = 60 % (6 out of 10 Hello packets are processed by Babel, 4 our discarded immediately)
- RSS = -105 dBm, MSE = -9 dB
  - MSE is below the Hard threshold = Hello packet is discarded

Let's imagine we have one unit and it has at least two neighbors (10.10.10.2 and 10.10.10.3 are the neighbors' Radio IP addresses). For the neighbor '2' the modulation is set to 64QAM, for the neighbor '3' the modulation is '256QAM'. We set two individual thresholds, e.g.:

- Link to 10.10.10.2
  - RSS
    - Soft: -72 dBm
    - Hard: -92 dBm
  - MSE
    - Soft: -27 dB
    - Hard: -22 dB
- Link to 10.10.10.3
  - RSS
    - Soft: -68 dBm
    - Hard: -88 dBm
  - MSE
    - Soft: -33 dB
    - Hard: -28 dB

[1] https://www.racom.eu/eng/products/m/ripex2/tech.html#tech-det
[2] https://www.racom.eu/download/hw/ripex/free/eng/1_application/ripex2-app-mse-en.pdf

Further optimization can be discussed with our technical support team via *support@racom.eu*[3].

Within the example topology we utilize 25kHz channel and 16DEQAM modulation. We do not set any Individual thresholds, in case you need them, check the information above. Default thresholds in all four units will be set to

- RSS
  - Soft: -78 dBm
  - Hard: -98 dBm
- MSE
  - Soft: -22 dB
  - Hard: -17 dB

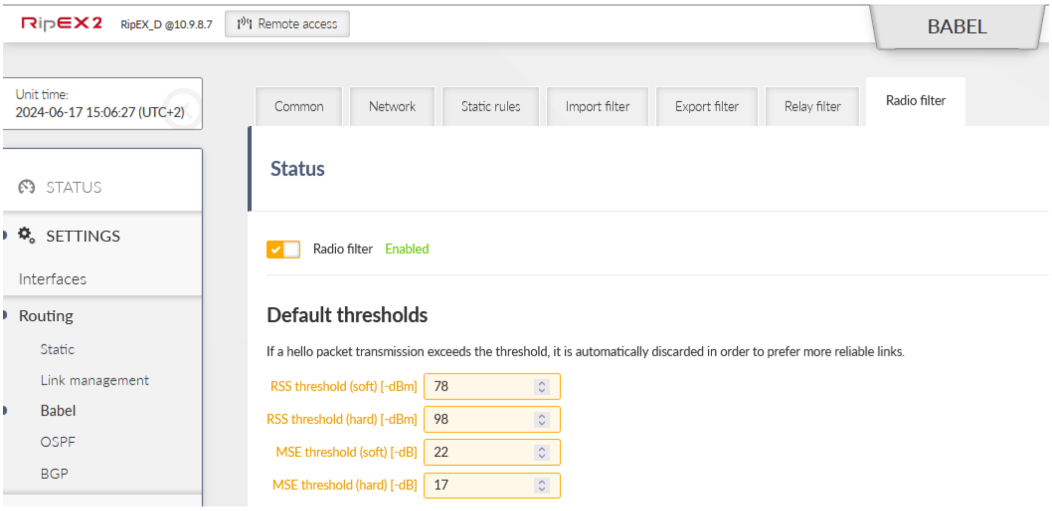


Fig. 2.1: Default Radio filter thresholds for 16DEQAM

Save this change in all 4 units.

[3] mailto:support@racom.eu

Check the Babel Status after a while. Based on your conditions, you may see similar output.

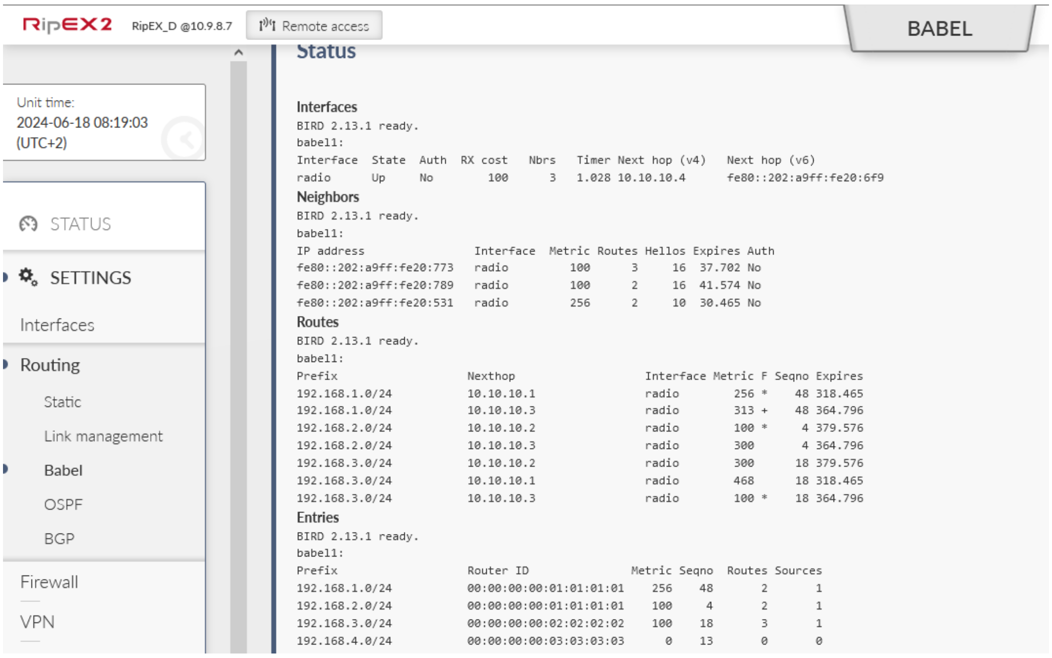


Fig. 2.2: RipEX_D Babel routing

Within our example output, signals to 10.10.10.2 and 10.10.10.3 are better than configured thresholds, but signals to 10.10.10.1 are between the soft and hard thresholds, i.e. we only got 10 out of 16 Hello packets correctly and the direct metric is now 256. It is still the lowest metric so the direct radio link is used.

Based on Statistics, MSE values are better than the soft threshold, but signals to 10.10.10.1 are in average -83 dBm which is close to soft threshold, but worse. The minimum measured signal is even -89 dBm. This is the reason why some of the Hello packets are discarded and the metric is higher.

Radio signal statistics

Link address	IP address	Header count	RSS [dBm]			
			avg	dev	min	max
:20:05:31	10.10.10.1	167	-83	2	-89	-77
:20:07:73	10.10.10.3	190	-79	1	-82	-76
:20:07:89	10.10.10.2	162	-76	0	-78	-76

Fig. 2.3: RipEX_D Radio signal statistics

Discarding is based on probability so once you check the Status again, it can be different, e.g.:

```
Neighbors
BIRD 2.13.1 ready.
babel1:
IP address               Interface  Metric Routes Hellos Expires Auth
fe80::202:a9ff:fe20:773  radio         133      3     12  30.111 No
fe80::202:a9ff:fe20:789  radio         100      2     16  33.985 No
fe80::202:a9ff:fe20:531  radio         177      2     12  22.874 No
Routes
BIRD 2.13.1 ready.
babel1:
Prefix                   Nexthop                    Interface Metric F Seqno Expires
192.168.1.0/24           10.10.10.1                 radio        177 *    48 310.875
192.168.1.0/24           10.10.10.3                 radio        285      48 357.206
192.168.2.0/24           10.10.10.2                 radio        100 *     4 371.985
192.168.2.0/24           10.10.10.3                 radio        361       4 357.206
192.168.3.0/24           10.10.10.2                 radio        328      18 371.985
192.168.3.0/24           10.10.10.1                 radio        329      18 310.875
192.168.3.0/24           10.10.10.3                 radio        133 *    18 357.206
```

Fig. 2.4: RipEX_D Babel routing, #2

Important

Individual thresholds can be taken into account only if the IP address is known and correct Link address and IP address translation has been done. Check your Statistics. For each remote you only see 0.0.0.0 as the IP address, the default thresholds are used. Once there is any communication with a particular IP address, ARP data are exchanged and the IP address is known afterwards. In case there is regular traffic with all the neighbors (SCADA, …), the IP addresses are always known, but there might be multiple other sites within the radio coverage (visible in Statistics), but with no user traffic resulting in only Link addresses within Statistics and IPs being 0.0.0.0.

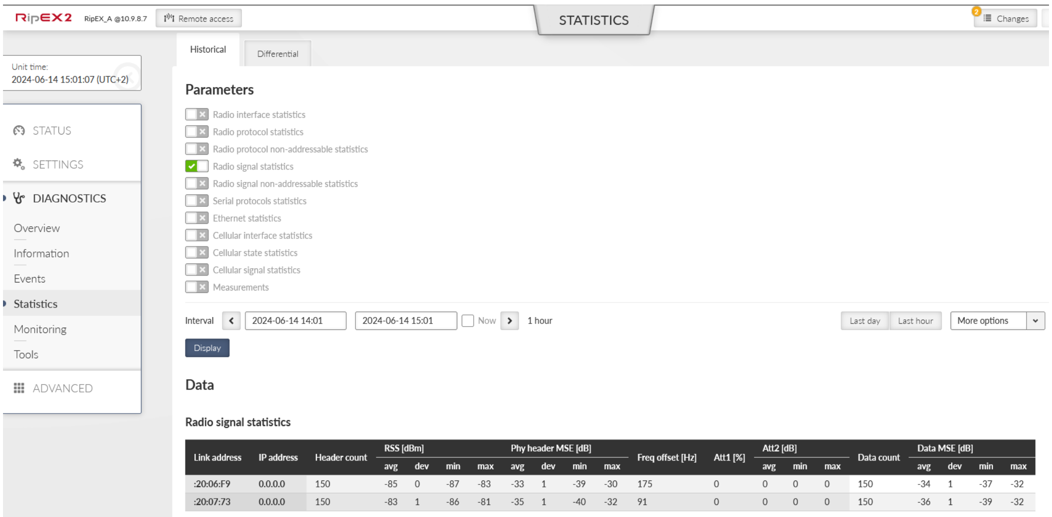


Fig. 2.5: Unknown IP addresses for Link addresses

2.2. Relay filters

Relay filters can be used for

- Setting any unit to be the end station not forwarding any obtained/received routes from its neighbours
- Increasing metric for routes being propagated – i.e. to disadvantage paths leading through this unit within the Babel dynamic routing (higher routing rule metric via this unit)
- Limiting number of propagated routing rules – i.e. decrease the number of routes within the network

In case the unit is not supposed to propagate any obtained routes further, so it serves as the end-station (terminal, not repeater), you can reject all rules to be propagated/relayed.

Just go to the SETTINGS > Routing > Babel > Relay filter menu and change the policy to “Reject”.

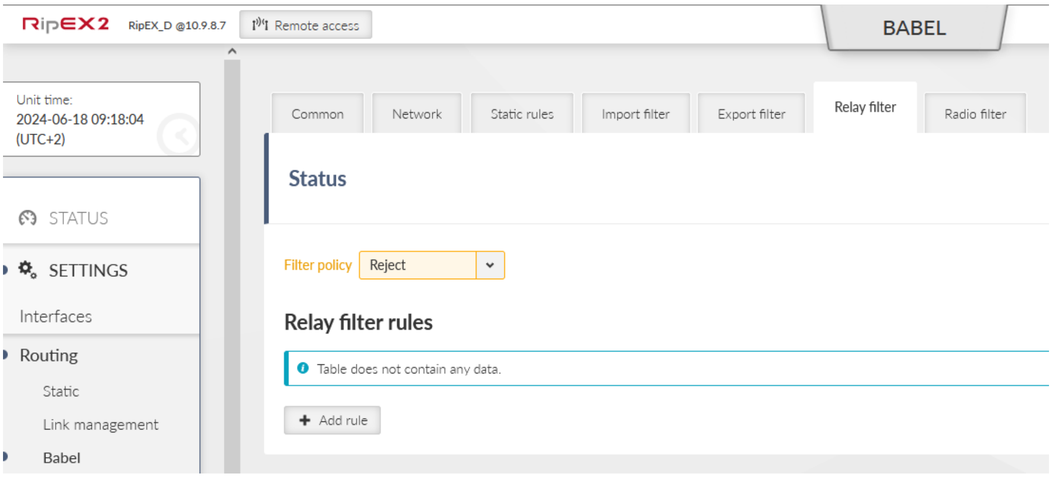


Fig. 2.6: RipEX_D filter policy set to “Reject”

Until this change, e.g. the RipEX_A could have similar Babel routes.

Routes

```
BIRD 2.13.1 ready.
babel1:
Prefix               Nexthop              Interface Metric F Seqno Expires
192.168.3.0/24       10.10.10.3           radio        246 *    20 356.948
192.168.3.0/24       10.10.10.4           radio        383 +    20 324.205
192.168.3.0/24       10.10.10.2           radio        428 +    20 371.729
192.168.4.0/24       10.10.10.3           radio        346 +    14 356.948
192.168.4.0/24       10.10.10.4           radio        283 *    14 324.205
192.168.4.0/24       10.10.10.2           radio        328 +    14 371.729
```

Fig. 2.7: RipEX_A Babel routes

We can see RipEX_D (10.10.10.4) is also used as a gateway for 192.168.3.0/24 (RipEX_C). If we set that Reject policy in RipEX_D, no rule will not be propagated from RipEX_D to other units except its own configured subnets.

```
Routes
BIRD 2.13.1 ready.
babel1:
Prefix                  Nexthop                   Interface Metric F Seqno Expires
192.168.3.0/24          10.10.10.3                radio        314 *    21 357.779
192.168.3.0/24          10.10.10.2                radio        580      21 372.560
192.168.4.0/24          10.10.10.3                radio        444 +    14 357.779
192.168.4.0/24          10.10.10.4                radio        141 *    14 325.035
192.168.4.0/24          10.10.10.2                radio        274 +    14 372.560
```

Fig. 2.8: RipEX_A Babel routes after setting RipEX_D to reject relay policy

You can also increase metric for the routes being propagated, e.g. to disadvantage particular repeater. Let’s set RipEX_C (10.10.10.3) to increase metrics obtained from this repeater by 100.

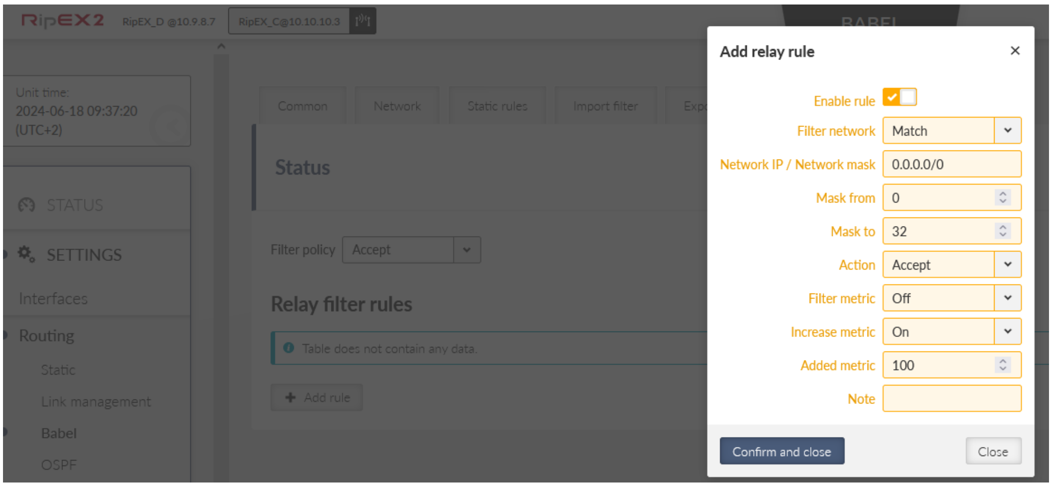


Fig. 2.9: RipEX_C Relay filter to increase metric by 100

We can see similar output in neighbouring units, e.g. from RipEX_D.

```
Routes
BIRD 2.13.1 ready.
babel1:
Prefix                  Nexthop                   Interface Metric F Seqno Expires
192.168.1.0/24          10.10.10.1                radio        100 *    52 311.901
192.168.1.0/24          10.10.10.3                radio        333 +    52 357.797
192.168.2.0/24          10.10.10.2                radio        100 *     6 373.014
192.168.2.0/24          10.10.10.3                radio        300 +     6 357.797
192.168.3.0/24          10.10.10.2                radio        200 +    23 373.014
192.168.3.0/24          10.10.10.1                radio        233 +    23 311.901
192.168.3.0/24          10.10.10.3                radio        100 *    23 357.797
```

Fig. 2.10: RipEX_D Babel routes with higher metrics via 10.10.10.3

Routes to 192.168.1.0/24 and 192.168.2.0/24 via 10.10.10.3 have the highest metric. Direct route to its own subnet 192.168.3.0/24 is not disadvantaged this way, as required/configured. If you need to increase the metric of this direct path as well, do it via Static rules tab of Babel menu and increase the “Metric” parameter for the given Destination IP/mask.

Last, but not least, is to limit just some of the propagated routes, not all, via the “Reject” Relay filter policy. Go to the RipEX_D again and change the Policy to “Accept”, but let’s reject propagating 192.168.2.0/24 and 192.168.3.0/24 networks. I.e. the unit will only propagate 192.168.1.0/24 (and of course its own subnet, but this not controlled by the Relay filter).

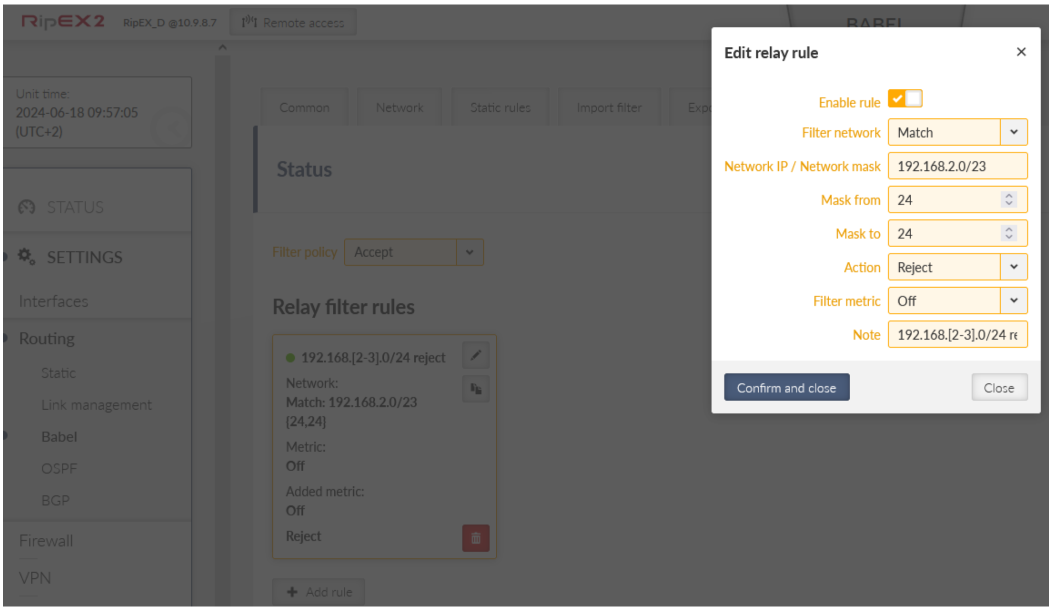


Fig. 2.11: RipEX_D Relay filter – rejecting 192.168.2.0/23 subnets with /24 mask

We know we should receive 192.168.2.0/24 and 192.168.3.0/24, i.e. /24 mask in both cases. We can limit the Mask from/to parameters to suit our scenario. We could also have two individual rules or do it via a single rule using /23 mask.

The default policy is “accept” so the 192.168.1.0/24 network will be propagated.

Routes

```
BIRD 2.13.1 ready.
babel1:
Prefix                    Nexthop                    Interface Metric F Seqno Expires
192.168.1.0/24            10.10.10.1                 radio        200 +    52 361.851
192.168.1.0/24            10.10.10.4                 radio        200 *    52 375.410
192.168.2.0/24            10.10.10.2                 radio        100 *     6 302.427
192.168.4.0/24            10.10.10.4                 radio        100 *    14 375.410
192.168.4.0/24            10.10.10.1                 radio        300      14 361.851
192.168.4.0/24            10.10.10.2                 radio        200      14 302.427
```

Fig. 2.12: RipEX_C Babel routes without 10.10.10.4 for 192.168.[2-3].0/24 networks

RipEX_C does not use 10.10.10.4 as a gateway to reach 192.168.2.0/24 and 192.168.3.0/24 networks. Obviously, it uses the gateway to reach 192.168.1.0/24 and 192.168.4.0/24.

Note

If you configure Babel routing for the fist time in your network, it is suggested to reboot each device after final configuration changes so the dynamic protocol is fully restarted in all units with correct and current configuration.

Sure, you can combine all the options in every unit within the whole network to suit your requirements.

Further optimization can be discussed with our technical support team via *support@racom.eu*[4].

[4] mailto:support@racom.eu

3. Two repeaters on the RF channel

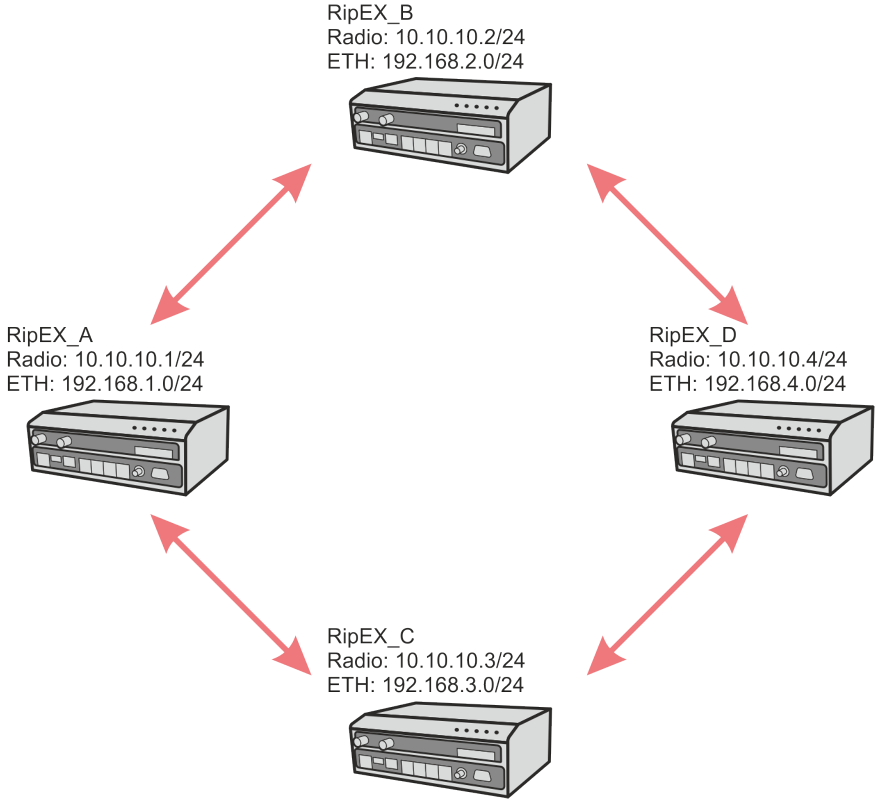


Fig. 3.1: Example 3 – Two repeaters topology

3.1. Description and Configuration

The third example is very similar to the first one, but the difference is that RipEX_A and RipEX_D can only communicate via repeaters and repeaters do not “see” each other. This is a typical situation in the field. For the test in an office, we use a frequency pair so that RipEX_A and RipEX_D send data on frequency 415.5 MHz and receive on 436 MHz, whereas RipEX units B and C have it vice versa.

Otherwise, the setup is completely the same.

If RipEX_A and RipEX_D are about to communicate only to each other, it might be useful to turn off "Routing offering".

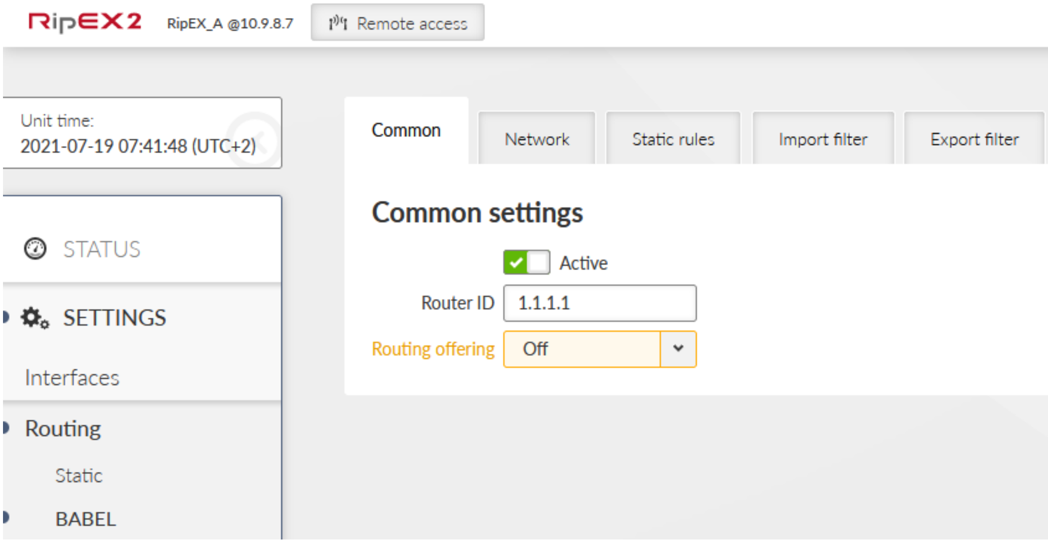


Fig. 3.2: Routing offering

That would limit forwarding received routes from units A and D. Thus, RipEX_B and C will not be able to reach each other and there will be less Babel routes in all devices, you can compare it within this example. We will enable full routing so "Routing offering" will be "on" in all four units.

Change the frequencies in all units accordingly. Antenna configuration can be set to Single or Dual, it does not change the behaviour, neither performance while testing with dummy loads on your desk.

RipEX_A, RipEX_D

- TX: 415.500.000 MHz
- RX: 436.000.000 MHz

RipEX_B, RipEX_C

- TX: 436.000.000 MHz
- RX: 415.500.000 MHz

3.2. Diagnostics and Testing

Check the Routing in Diagnostics menu. All three routes should be accessible via both repeaters 10.10.10.2 and 10.10.10.3 with the same price. One route is selected to be used; another one is a failure (candidate) option.

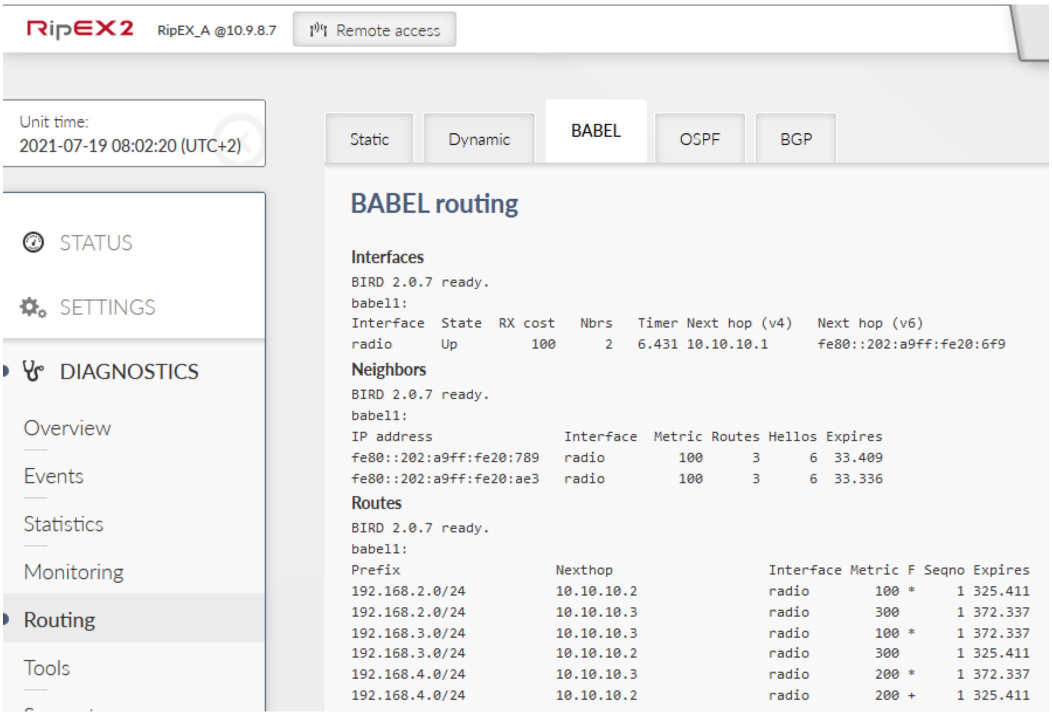


Fig. 3.3: Babel Routing state

If you want to prioritize one of the repeaters, you can do it by decreasing the cost in this repeater. E.g., decrease it to 10 and check the routing again.

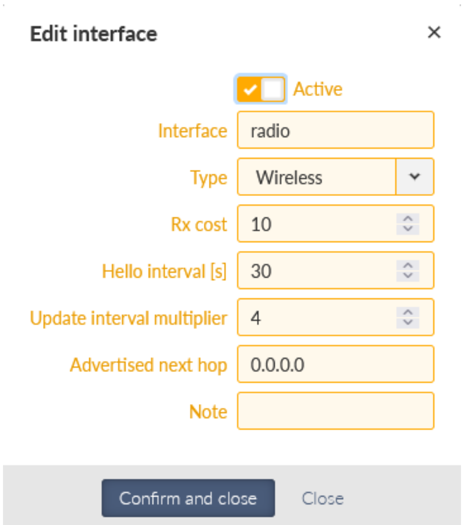


Fig. 3.4: Rx cost decreased in RipEX_C

Neighbors

```
BIRD 2.0.7 ready.
babel1:
IP address                Interface  Metric Routes Hellos Expires
fe80::202:a9ff:fe20:789   radio         100      3     16  18.264
fe80::202:a9ff:fe20:ae3   radio          10      3     16  17.944
```

Routes

```
BIRD 2.0.7 ready.
babel1:
Prefix                   Nexthop                  Interface Metric F Seqno Expires
192.168.2.0/24           10.10.10.2               radio        100 *     1 340.265
192.168.2.0/24           10.10.10.3               radio        210       1 386.947
192.168.3.0/24           10.10.10.3               radio         10 *     1 386.947
192.168.3.0/24           10.10.10.2               radio        300       1 340.265
192.168.4.0/24           10.10.10.3               radio        124 *     1 386.947
192.168.4.0/24           10.10.10.2               radio        200 +     1 340.265
```

Entries

```
BIRD 2.0.7 ready.
babel1:
Prefix                   Router ID               Metric Seqno  Routes Sources
192.168.1.0/24           00:00:00:00:01:01:01:01      0     1       0       0
192.168.2.0/24           00:00:00:00:02:02:02:02    100     1       2       1
192.168.3.0/24           00:00:00:00:03:03:03:03     10     1       2       1
192.168.4.0/24           00:00:00:00:04:04:04:04    124     1       2       1
```

Fig. 3.5: Changed routing costs

As you can see, route to 192.168.3.0/24 has a metric “10” now. Also, a route to 192.168.4.0/24 is 124, it could be 110, but some Hello packets were lost and it is increased for a while. Original values were 100 and 200.

And of course, Nexthop gateway is set to 10.10.10.3 for both routes.

3.2.1. Tools

Now, start a ping from RipEX_A to RipEX_D. Set a count to 1000.

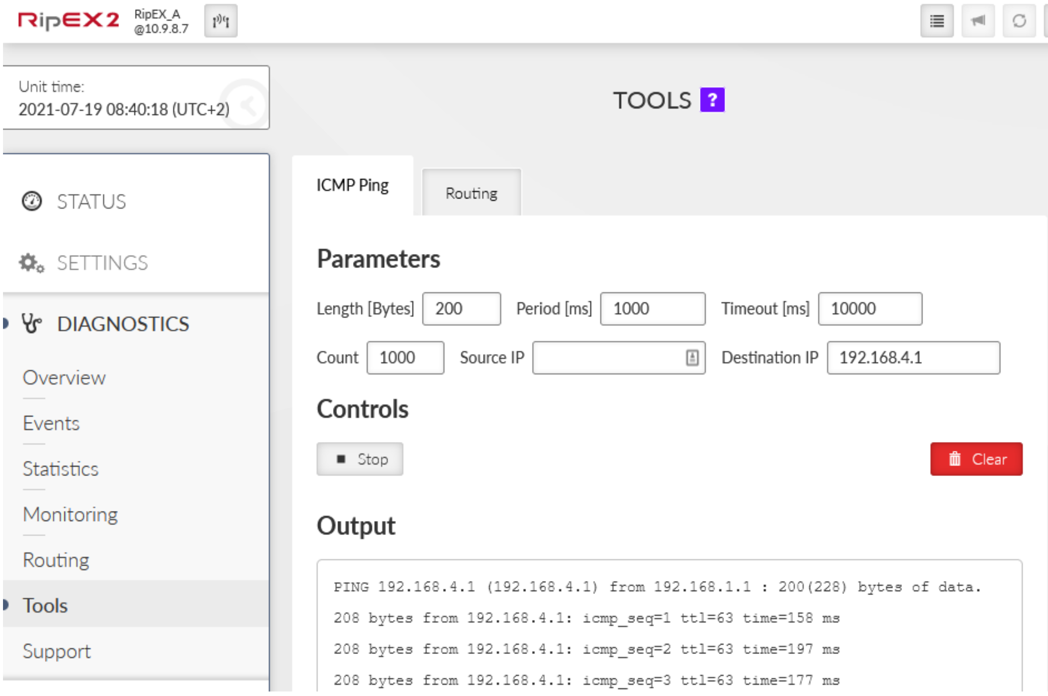


Fig. 3.6: RipEX_A – Ping to 192.168.4.1 (RipEX_D)

In another browser panel/window, open RipEX_A GUI again and check the Routing diagnostics. You can keep clicking on the “Refresh” button to see current data.

Turn off RipEX_C (currently used RipEX as a repeater for traffic). You will stop seeing successful pings, but after some time, you should start seeing correct replies (via RipEX_B). You should also see metrics being increased in the Routing diagnostics menu.

Note that even working routes can have higher metric due to propagating forwarded routes from “end devices”. If you had disabled RipEX_A and RipEX_D “routing offering” option, metrics would be more stable.

The last option is to use a tool called “**RSS ping**” (since RipEX2 firmware 2.0.5.0). RSS ping is a diagnostic tool for the radio performance measurement (RSS and MSE) of the individual radio hops within a RipEX2 network. RSS ping sends UDP data on port 8906. You should see changes in used repeater on the end to end communication path.

```
Neighbors
BIRD 2.0.7 ready.
babel1:
IP address                 Interface  Metric Routes Hellos Expires
fe80::202:a9ff:fe20:789    radio         100      3     16  28.385
fe80::202:a9ff:fe20:ae3    radio       65535      3      6  28.078
Routes
BIRD 2.0.7 ready.
babel1:
Prefix                    Nexthop                    Interface Metric F Seqno Expires
192.168.2.0/24            10.10.10.2                 radio        100 *     1 320.388
192.168.2.0/24            10.10.10.3                 radio      65535       1   7.078
192.168.3.0/24            10.10.10.3                 radio      65535 +     1   7.078
192.168.3.0/24            10.10.10.2                 radio        213       1 320.388
192.168.4.0/24            10.10.10.3                 radio      65535 +     1   7.078
192.168.4.0/24            10.10.10.2                 radio        214       1 320.388
```

Fig. 3.7: RipEX_C turned off, changes in metrics

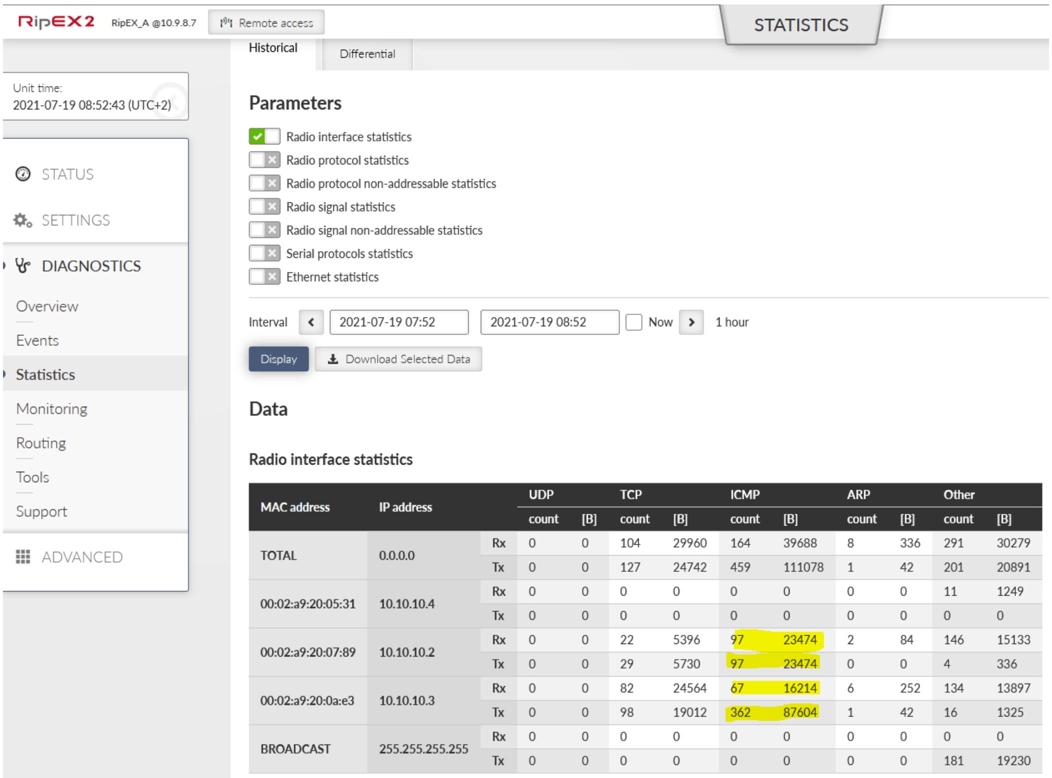
```
208 bytes from 192.168.4.1: icmp_seq=20 ttl=63 time=177 ms
208 bytes from 192.168.4.1: icmp_seq=21 ttl=63 time=164 ms
208 bytes from 192.168.4.1: icmp_seq=22 ttl=63 time=171 ms
ping: sendmsg: No route to host
ping: sendmsg: No route to host
208 bytes from 192.168.4.1: icmp_seq=320 ttl=63 time=297 ms
208 bytes from 192.168.4.1: icmp_seq=321 ttl=63 time=158 ms
208 bytes from 192.168.4.1: icmp_seq=322 ttl=63 time=210 ms
```

Fig. 3.8: RipEX_A – ping

As you can see, it took 5 minutes to Babel to increase the metric high enough so the backup repeater is used. It is mainly due to low metric “10” of RipEX_C setup. If you change it to 50, it will be quicker (its metric goes up faster).

Turn on RipEX_C again and check the Routing again. Note you can also check Monitoring menu for live traffic monitoring.

One another option where to check how this works are Statistics. You should see some ICMP data being sent to 10.10.10.3 and some to 10.10.10.2. And there should be more Tx than Rx data for RipEX_C (10.10.10.3), because it was turned off for a while.



MAC address	IP address		UDP		TCP		ICMP		ARP		Other	
			count	[B]	count	[B]	count	[B]	count	[B]	count	[B]
TOTAL	0.0.0.0	Rx	0	0	104	29960	164	39688	8	336	291	30279
		Tx	0	0	127	24742	459	111078	1	42	201	20891
00:02:a9:20:05:31	10.10.10.4	Rx	0	0	0	0	0	0	0	0	11	1249
		Tx	0	0	0	0	0	0	0	0	0	0
00:02:a9:20:07:89	10.10.10.2	Rx	0	0	22	5396	97	23474	2	84	146	15133
		Tx	0	0	29	5730	97	23474	0	0	4	336
00:02:a9:20:0a:e3	10.10.10.3	Rx	0	0	82	24564	67	16214	6	252	134	13897
		Tx	0	0	98	19012	362	87604	1	42	16	1325
BROADCAST	255.255.255.255	Rx	0	0	0	0	0	0	0	0	0	0
		Tx	0	0	0	0	0	0	0	0	181	19230

Fig. 3.9: RipEX_A – Statistics (Radio interface statistics)

4. Radio channel and Ethernet combination

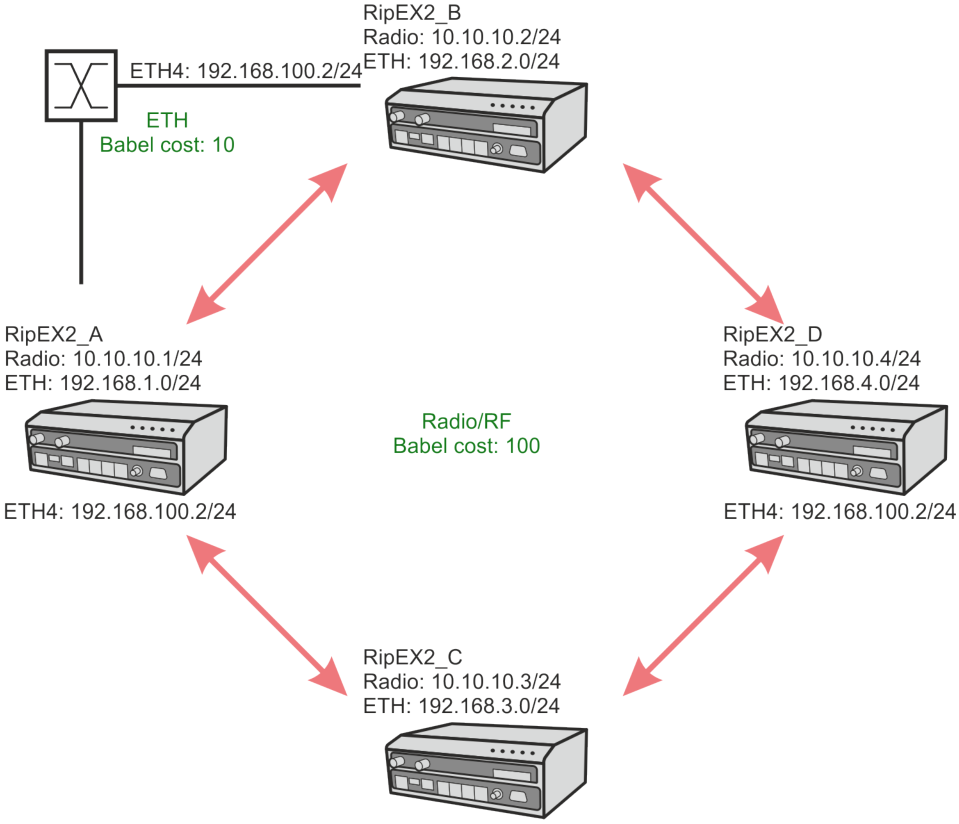


Fig. 4.1: Example 4 – Radio channel and Ethernet combination

4.1. Description

The fourth example requires you to have an additional one simple switch and two Ethernet cables. Interconnect RipEX_A and RipEX_B via switch – plug the cable into their ETH4 ports.

Note

It is also possible to interconnect RipEX2 devices directly with a cable only. External switch is not mandatory.

Radio/RF channel cost stay the same for all units, set to 100. The Ethernet path is on 100 Mbps link and Babel setup will utilize a low cost equal to 10 for this link and a faster Babel routes propagation, because the RF channel capacity is not an issue on Ethernet.

This example can show you how to utilize Babel if you have RipEX2 units connected to some fast “backbone” and would like to use RipEX2 link only as a backup; or to route particular radio segment via particular part of the backbone.

Note

As depicted, example 3 utilizes dual frequency solution from example 2.

4.2. RipEX_A Configuration

Start with ETH4 configuration. Go to the SETTINGS – Interfaces – Ethernet menu. Select ETH4 panel and detach it from a current bridge. Create a new interface called “backbone” and assign 192.168.100.1/24 IP address to it.

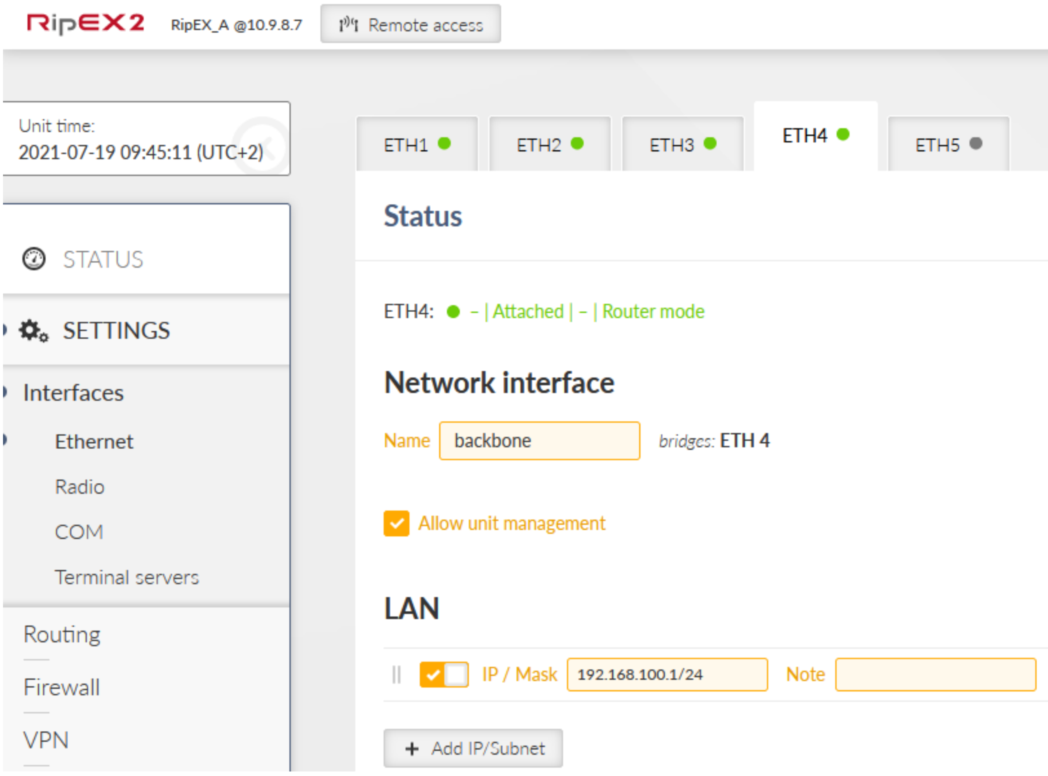


Fig. 4.2: RipEX_A – ETH4 configuration

The last menu is SETTINGS – Routing – Babel. Within the “Network” tab, add a new interface.

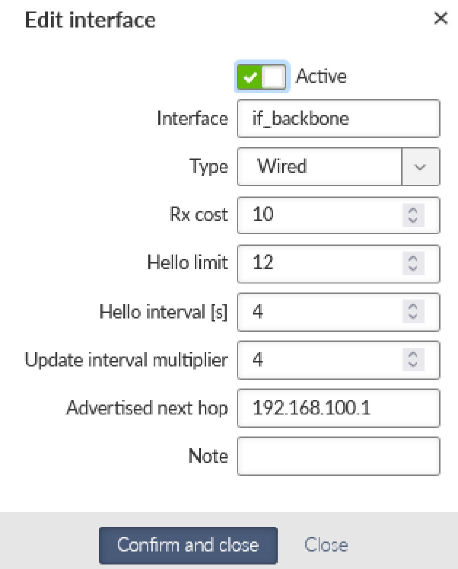


Fig. 4.3: RipEX_A – Babel setup (ETH)

Non-default parameters:

Interface	if_backbone
Type	Wired (Ethernet is used, not the RF channel)
Rx cost	10
Advertised next hop	192.168.100.1 (our ETH4 IP address)

Save the current configuration.

4.3. RipEX_B Configuration

A very similar setup is to be done in RipEX_B as well. Set the ETH4 IP address in a new “backbone” bridge to 192.168.100.2/24.

Note

You can optimize IP usage with /30 mask for “backbone” interface for our example.

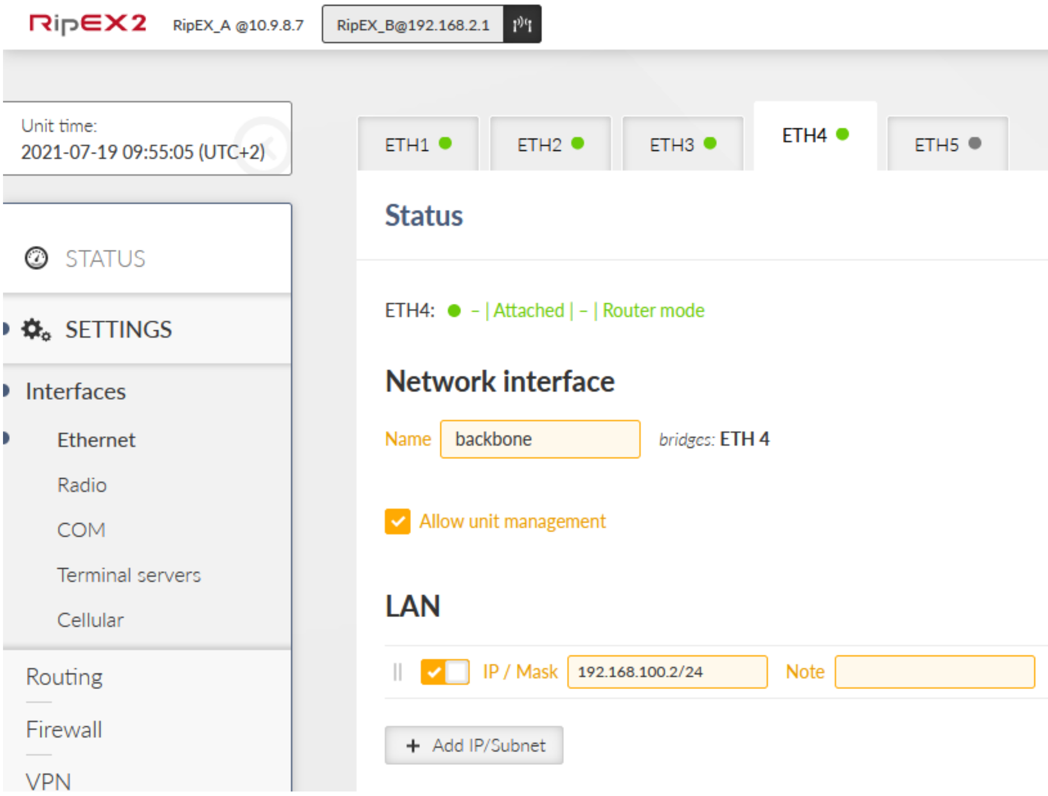


Fig. 4.4: RipEX_B – ETH4 setup

Create a new Babel Network interface.

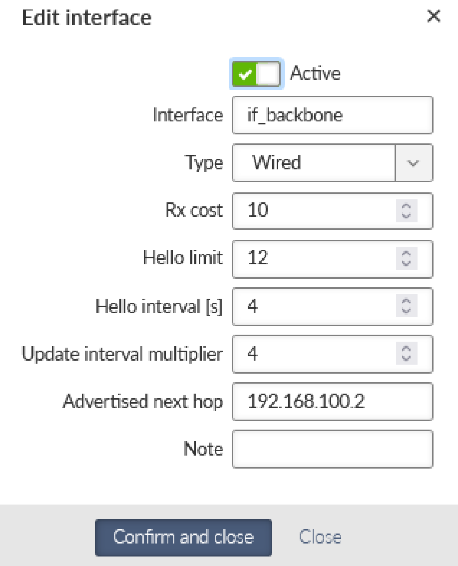


Fig. 4.5: RipEX_B – Babel setup (ETH)

Save the changes.

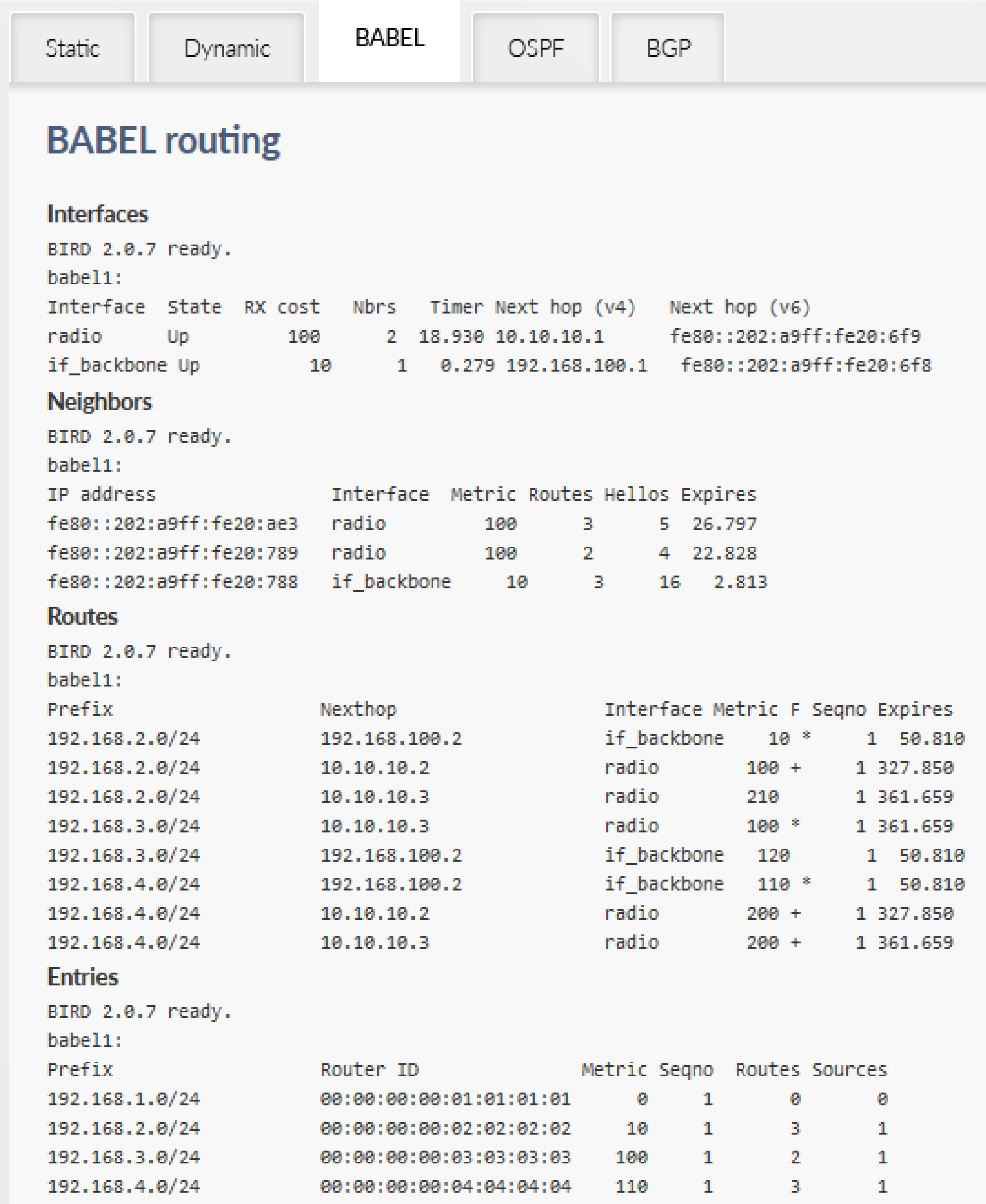
Important

Do not forget to increase the Cost in RipEX_C (192.168.3.1) back to 100 for the radio interface.

The configuration should be complete, there is no need to change other settings neither in RipEX_C, nor RipEX_D.

4.4. Diagnostics and Testing

Babel Routing in RipEX_A state can be similar to:


Static Dynamic BABEL OSPF BGP

BABEL routing

Interfaces

```
BIRD 2.0.7 ready.
babel1:
Interface  State  RX cost   Nbrs   Timer Next hop (v4)    Next hop (v6)
radio      Up         100      2  18.930 10.10.10.1      fe80::202:a9ff:fe20:6f9
if_backbone Up         10      1   0.279 192.168.100.1    fe80::202:a9ff:fe20:6f8
```

Neighbors

```
BIRD 2.0.7 ready.
babel1:
IP address                 Interface  Metric Routes Hellos Expires
fe80::202:a9ff:fe20:ae3    radio         100      3      5  26.797
fe80::202:a9ff:fe20:789    radio         100      2      4  22.828
fe80::202:a9ff:fe20:788    if_backbone    10      3     16   2.813
```

Routes

```
BIRD 2.0.7 ready.
babel1:
Prefix                   Nexthop                   Interface Metric F Seqno Expires
192.168.2.0/24           192.168.100.2             if_backbone   10 *      1  50.810
192.168.2.0/24           10.10.10.2                radio        100 +     1 327.850
192.168.2.0/24           10.10.10.3                radio        210       1 361.659
192.168.3.0/24           10.10.10.3                radio        100 *     1 361.659
192.168.3.0/24           192.168.100.2             if_backbone  120        1  50.810
192.168.4.0/24           192.168.100.2             if_backbone  110 *      1  50.810
192.168.4.0/24           10.10.10.2                radio        200 +     1 327.850
192.168.4.0/24           10.10.10.3                radio        200 +     1 361.659
```

Entries

```
BIRD 2.0.7 ready.
babel1:
Prefix                   Router ID               Metric Seqno  Routes Sources
192.168.1.0/24           00:00:00:00:01:01:01:01      0     1       0       0
192.168.2.0/24           00:00:00:00:02:02:02:02     10     1       3       1
192.168.3.0/24           00:00:00:00:03:03:03:03    100     1       2       1
192.168.4.0/24           00:00:00:00:04:04:04:04    110     1       3       1
```

Fig. 4.6: RipEX_A – Babel routing

Our radio can “see” neighbor on two interfaces, with cost of 10 and 100 and it has three neighbors in total (two neighbors are of RipEX_B).

To each remote subnet, we have three routes. The best one to 192.168.2.0/24 is, of course, via ETH with metric equal to just 10. We can get to 192.168.3.0/24 via one radio hop – metric is 100. And the last 192.168.4.0/24 subnet is accessible with metric equal to 110 (one hop ETH and one radio).

The last table shows four entries in Babel table – the first one is our RipEX_A and then, three neighboring units with particular Metric/cost.

4.4.1. Tools and Monitoring

As a test, you can (this time) run an ICMP ping test from your laptop connected via ETH1, ETH2 or ETH3 to RipEX_A. Ping a remote 192.168.4.1 RipEX2 ETH IP address. Do not forget to set a default route to 192.168.1.1 or set a static route to 192.168.4.0/24 via 192.168.1.1 on your laptop.

Your laptop should have any other IP address within 192.168.1.0/24 subnet, except .0 (network, .1 RipEX2 and .255 broadcast).

A command and its output from Windows CMD line can be similar to:

```
C:\Windows\system32>ping 192.168.4.1 -t
Pinging 192.168.4.1 with 32 bytes of data:
Reply from 192.168.4.1: bytes=32 time=61ms TTL=62
Reply from 192.168.4.1: bytes=32 time=48ms TTL=62
Reply from 192.168.4.1: bytes=32 time=42ms TTL=62
Reply from 192.168.4.1: bytes=32 time=48ms TTL=62
```

Verify that data are really sent via Ethernet. Run the Monitoring on ETH4 interface. Enable it, capture both Tx and Rx and limit the output to ICMP data only (All = “Off”, ICMP = “On”). Length can be 0 Bytes, because it is not important for us now. You should see similar ICMP output:

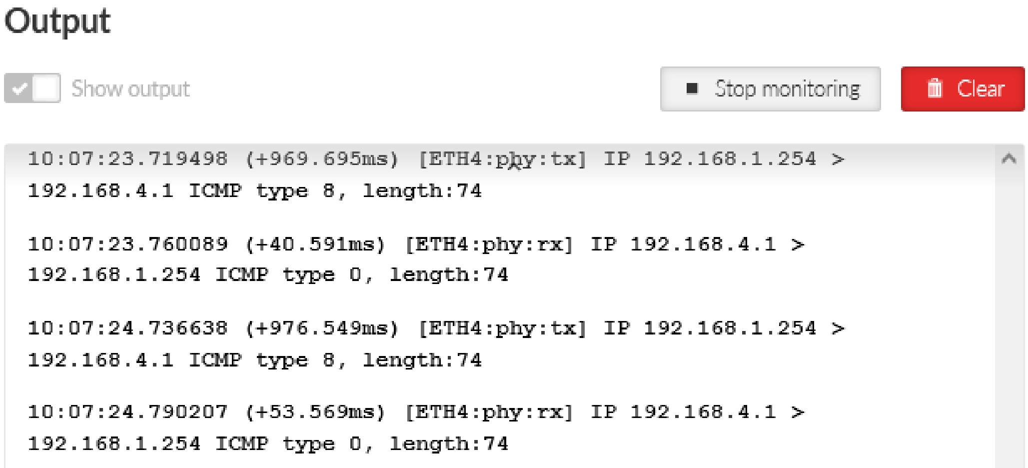


Fig. 4.7: RipEX_A – ETH monitoring

At the same time, enable monitoring on the Radio interface as well. Limit the output to ICMP data only (All = “Off”, ICMP = “On”). There should not be any data now. Unplug the ETH cable from RipEX_A or RipEX_B (the one attached to the switch) and check if data start going through the Radio interface.

```
15:11:57.168526 (+688.481ms) [RF:phy:tx] IP 192.168.1.254 >
192.168.4.1 ICMP type 8, length:78

15:11:57.248369 (+79.842ms) [RF:phy:rx] IP 192.168.4.1 > 192.168.1.254
ICMP type 0, length:78, rss:81 mse:24

15:11:58.184686 (+936.316ms) [RF:phy:tx] IP 192.168.1.254 >
192.168.4.1 ICMP type 8, length:78

15:11:58.270987 (+86.301ms) [RF:phy:rx] IP 192.168.4.1 > 192.168.1.254
ICMP type 0, length:78, rss:81 mse:29
```

Fig. 4.8: RipEX_A – Radio monitoring

Possible Babel routing while an Ethernet cable is disconnected:

Neighbors

```
BIRD 2.0.7 ready.
babel1:
IP address                Interface  Metric Routes Hellos Expires
fe80::202:a9ff:fe20:ae3   radio         123      3     16  15.357
fe80::202:a9ff:fe20:789   radio         114      3     16  41.390
fe80::202:a9ff:fe20:788   if_backbone  65535      3     11   1.374
```

Routes

```
BIRD 2.0.7 ready.
babel1:
Prefix                   Nexthop                   Interface Metric F Seqno Expires
192.168.2.0/24           192.168.100.2             if_backbone 65535 +     1  25.369
192.168.2.0/24           10.10.10.2                radio        114 *     1 406.411
192.168.2.0/24           10.10.10.3                radio        266       1 320.208
192.168.3.0/24           10.10.10.3                radio        123 *     6 320.208
192.168.3.0/24           192.168.100.2             if_backbone 65535       6  25.369
192.168.3.0/24           10.10.10.2                radio        328       6 406.411
192.168.4.0/24           192.168.100.2             if_backbone 65535 +     2  25.369
192.168.4.0/24           10.10.10.2                radio        228 +     2 406.411
192.168.4.0/24           10.10.10.3                radio        223 *     2 320.208
```

Fig. 4.9: RipEX_A – Babel routing with Ethernet being disconnected

If not completely gone, metric for GRE/ETH routes is quickly 65535 and thus, one of the Radio paths is used.

In the ETH4 Statistics, you should see a lot of IPv6 (Babel) traffic.

IPv4 other		IPv6	
count	[B]	count	[B]
1	46	14	1750
0	0	14	1280
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
936	92481	598	52061
991	96618	553	47687

Fig. 4.10: RipEX_A – Babel data on ETH4 interface

5. Radio channel and Cellular (LTE) combination

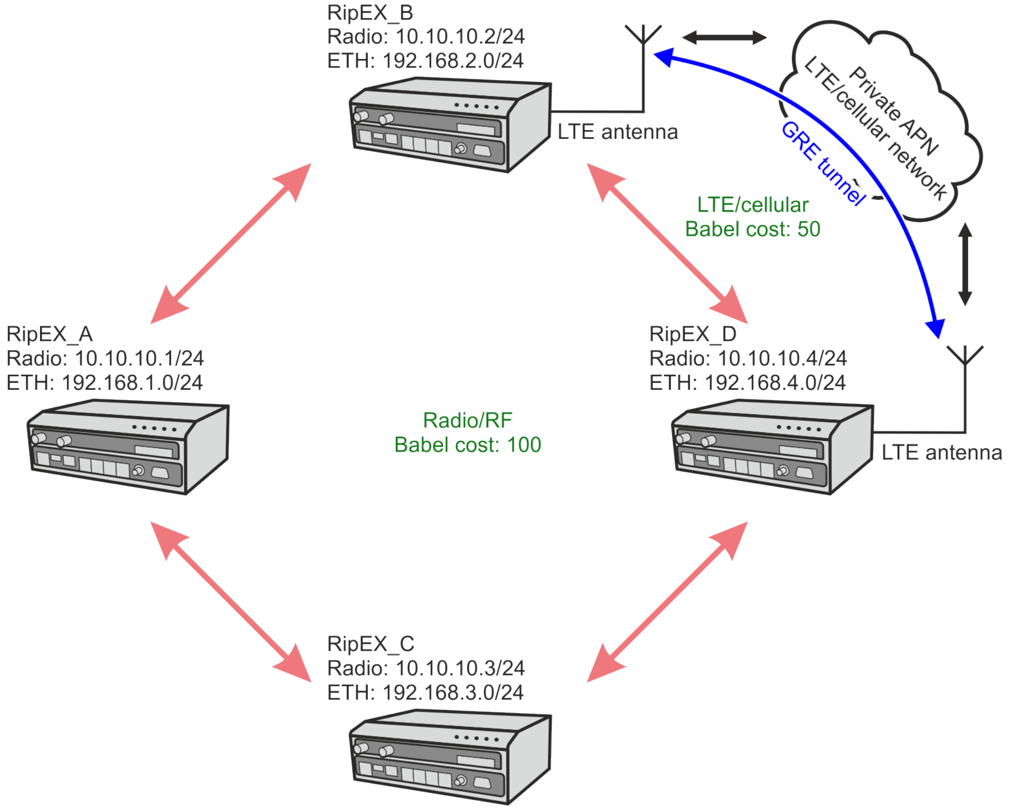


Fig. 5.1: Example 5 – Radio channel and Cellular LTE combination

5.1. Description

The fifth example depicts a situation with two RipEX2 units equipped with LTE module. Both units are connected to the private APN and GRE tunnel is built between them to enable mutual communication over the cellular network.

Note

Check M!DGE2 application notes and manual for more details about cellular networks and APN explanations.

This cellular path will be set with Rx cost equal to 50 to be better than RF channel, but worse than Ethernet connection. It can resemble Wired networks, but we will use Wireless. Choose whatever fits you more in your particular installation.

The example uses the third example configuration as a startup point.

5.2. RipEX_B Configuration

Important

Keep in mind we set up RACOM private APN. Your APN setup and IPs will be different! Make appropriate changes so that your configuration reflects your APN details.

Enable the cellular interface.

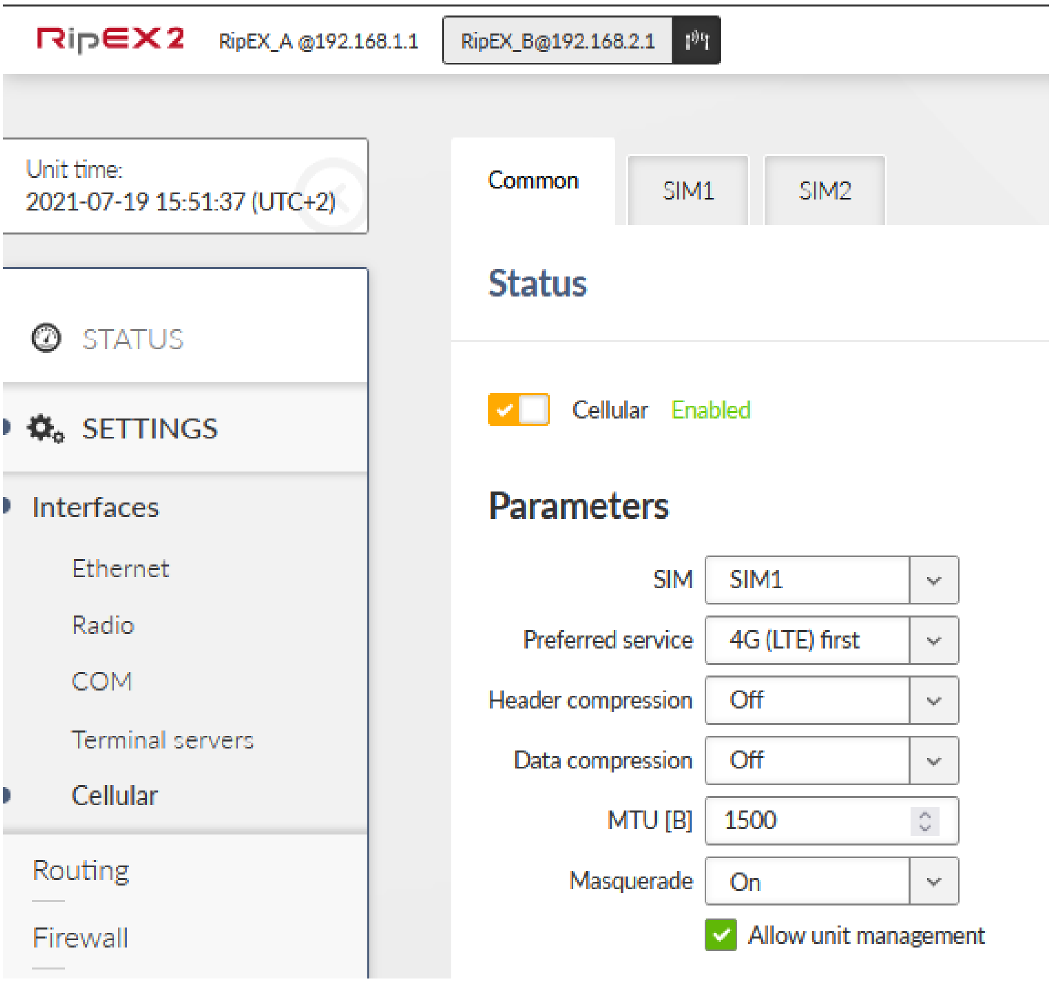


Fig. 5.2: RipEX_B – Cellular interface

You can keep everything in defaults, but set it as required by you and the cellular operator.

Configure your APN details in SIM1 panel.

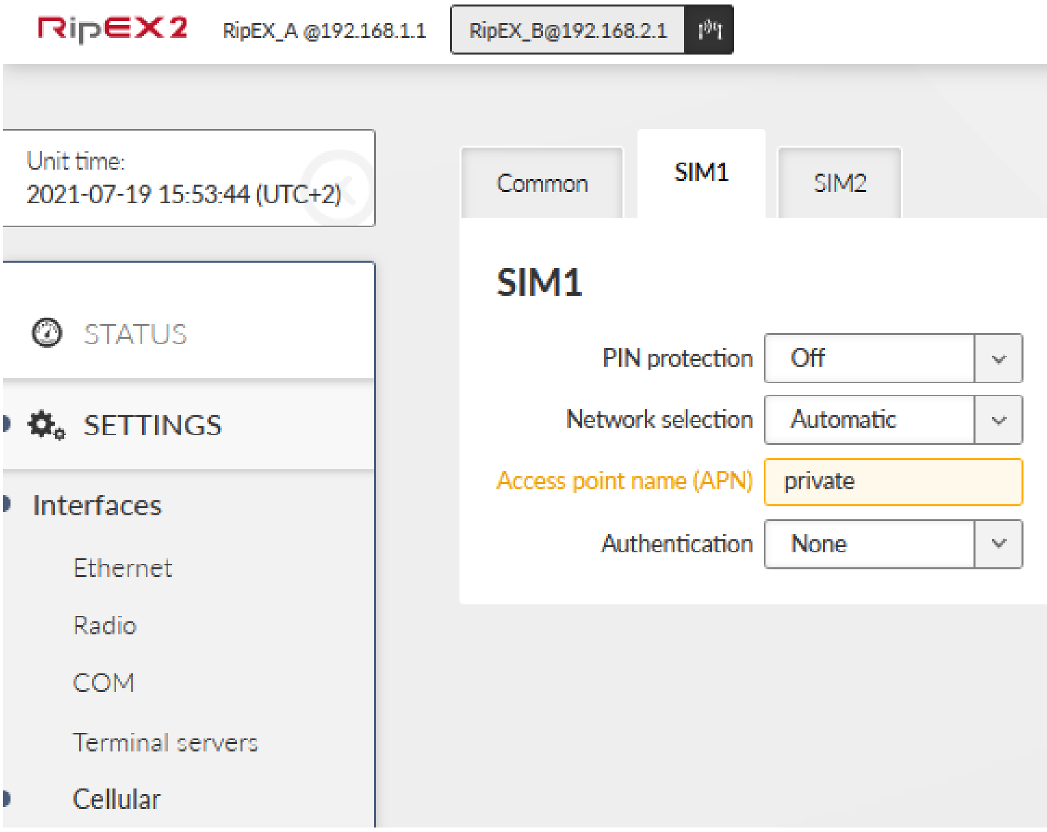


Fig. 5.3: RipEX_B – SIM1 setup

Create a GRE tunnel interface.

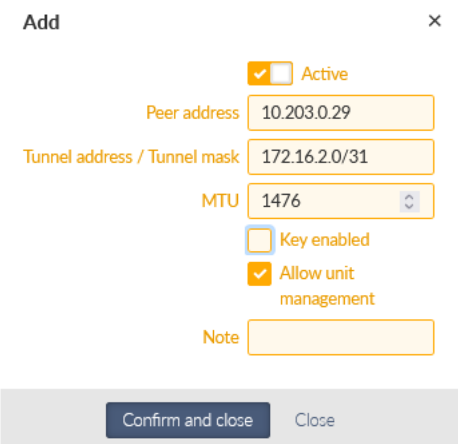


Fig. 5.4: RipEX_B – GRE tunnel over cellular network

The Peer IP address is the IP of the RipEX_D cellular interface. Tunnel address/mask is a new setup for GRE tunnel addresses.

The last step is configuring Babel Network interface again. Add a new interface.

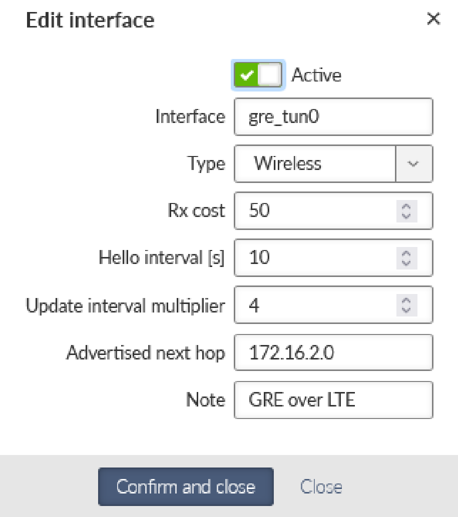


Fig. 5.5: RipEX_B – Babel interface (GRE over LTE)

Note the cellular interface is called “aux”, but we need to define a GRE interface. Its name is “gre_tun0”. Change the Type to “Wireless” and set the Rx cost to 50. The Hello interval parameter is increased from 4 to 10. It is still faster than over RF channel (30 seconds), but not so fast as via Ethernet (4 seconds). Optimize this number to suit your needs and possible data usage of your SIM cards.

The Advertised next hop is its local GRE IP address.

Do not forget to set a new static route over cellular/aux network. A new route should either be a default one, or like with our example, we route there APN’s subnet 10.203.0.0/17. Go to the SETTINGS – Routing – Static menu and add a new rule.

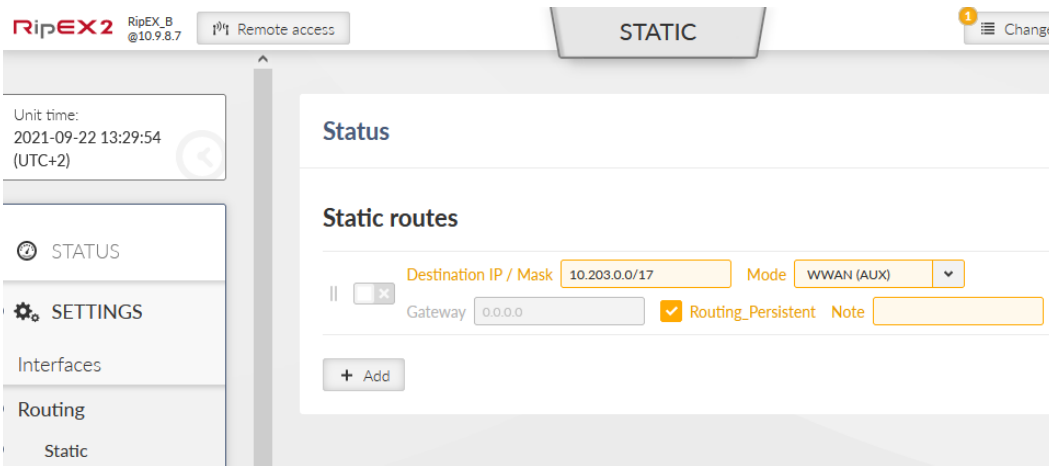


Fig. 5.6: RipEX_B – Static routing via cellular (aux) interface

Note that you can make this cellular/LTE routing rule permanent causing that if LTE is down, the rule is still active and traffic for LTE segment is not handled by another Routing rule (unreachable messages). Once LTE comes up again, the routing rule works as expected again.

To do so, check “Routing_Persistent” parameter in this routing rule. Save the changes.

5.3. RipEX_D Configuration

Common settings are the same. In your APN, you might have a different SIM1 setup compared to RipEX_B (each SIM can have different Authentication credentials).

Once the cellular interface is ready, create a new GRE tunnel setup again.

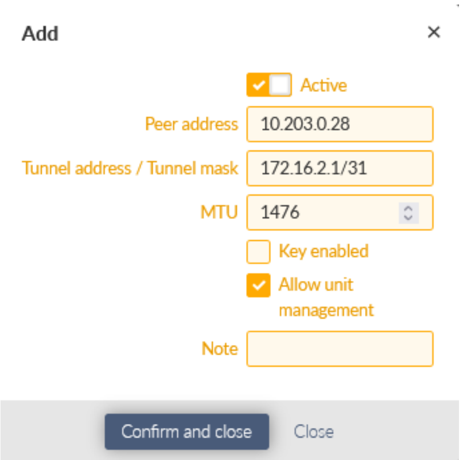


Fig. 5.7: RipEX_D – GRE tunnel setup

Peer address is the cellular IP of RipEX_B; 172.16.2.1/31 is a new local GRE IP address.

Add a new Babel Network interface.

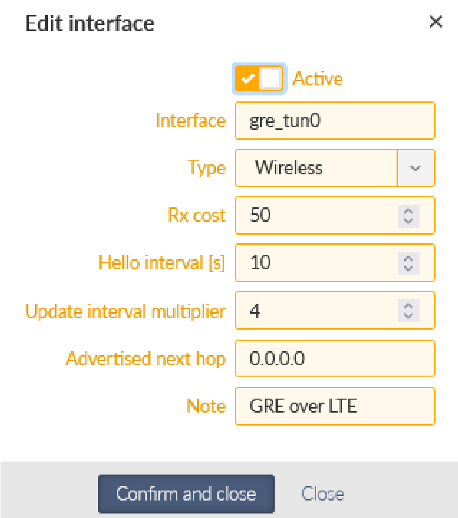


Fig. 5.8: RipEX_D – Babel interface (GRE over LTE)

Add a new static route via a cellular interface.

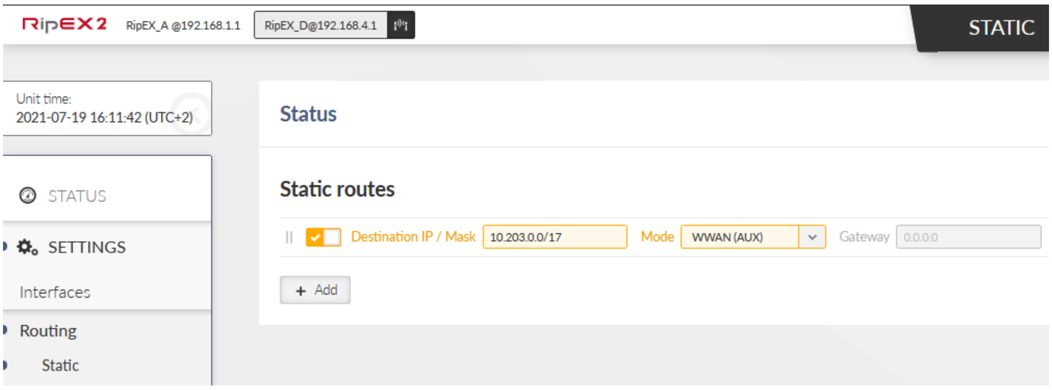


Fig. 5.9: RipEX_D – Static routing via cellular (aux) interface

You can make the rule persistent again.

Save the changes.

5.4. Diagnostics and Testing

Check the Cellular/LTE status in SETTINGS – Interfaces – Cellular menu.

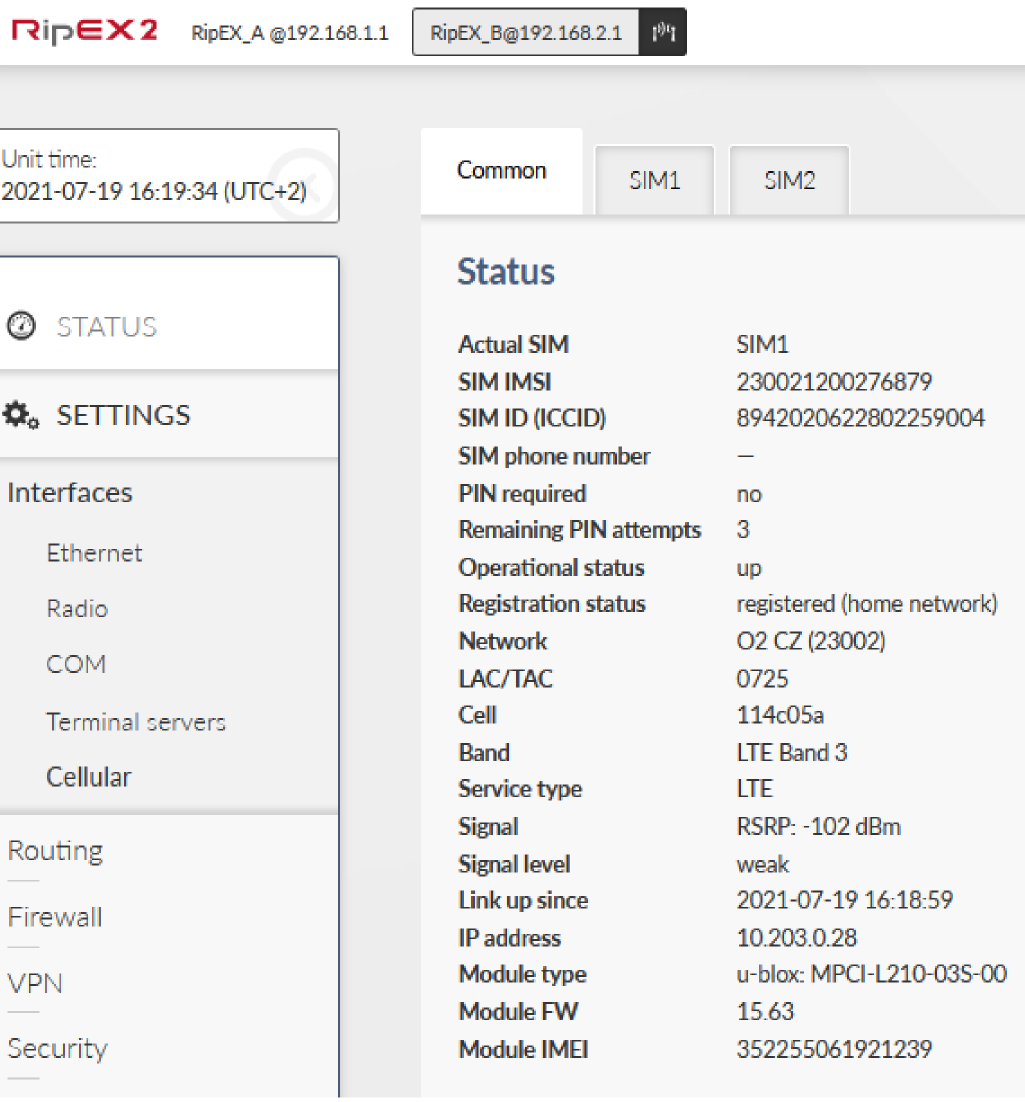


Fig. 5.10: RipEX_B – Cellular interface status

The status provides with a lot of information about current cellular connection.

Now, go to the Diagnostics and Tools menu. Try the mutual ping between 10.203.0.28 and 10.203.0.29. Also run a ping between 172.16.2.0 and 172.16.2.1 IP addresses.

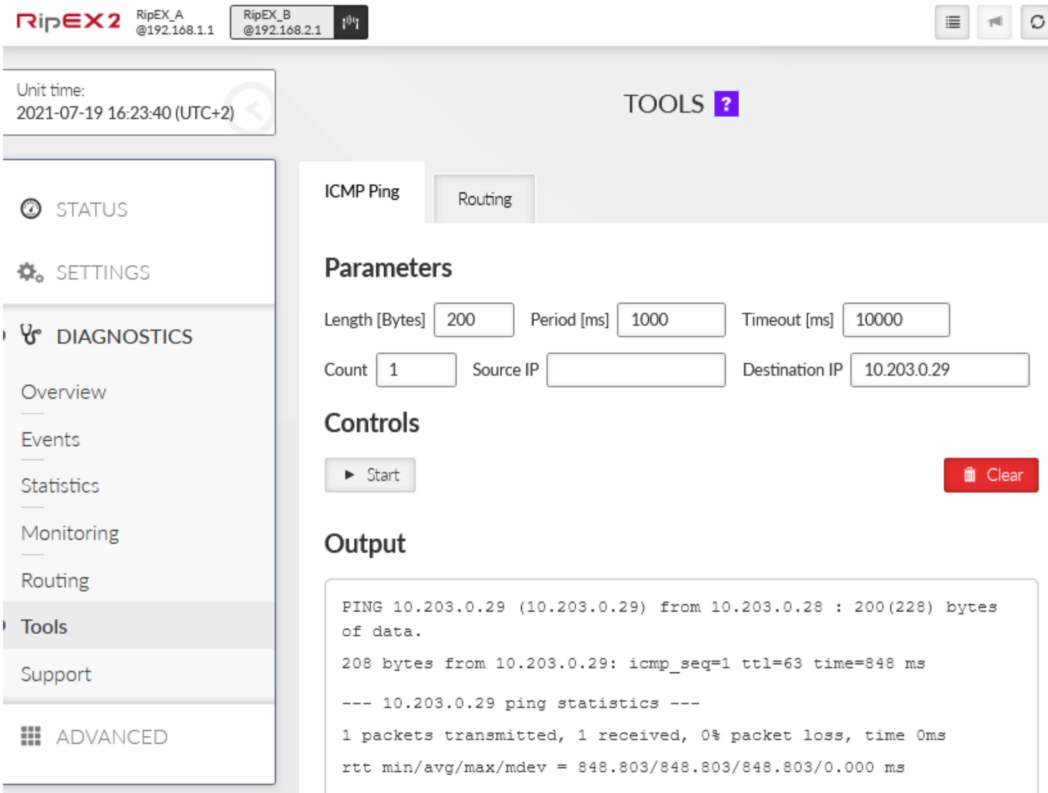
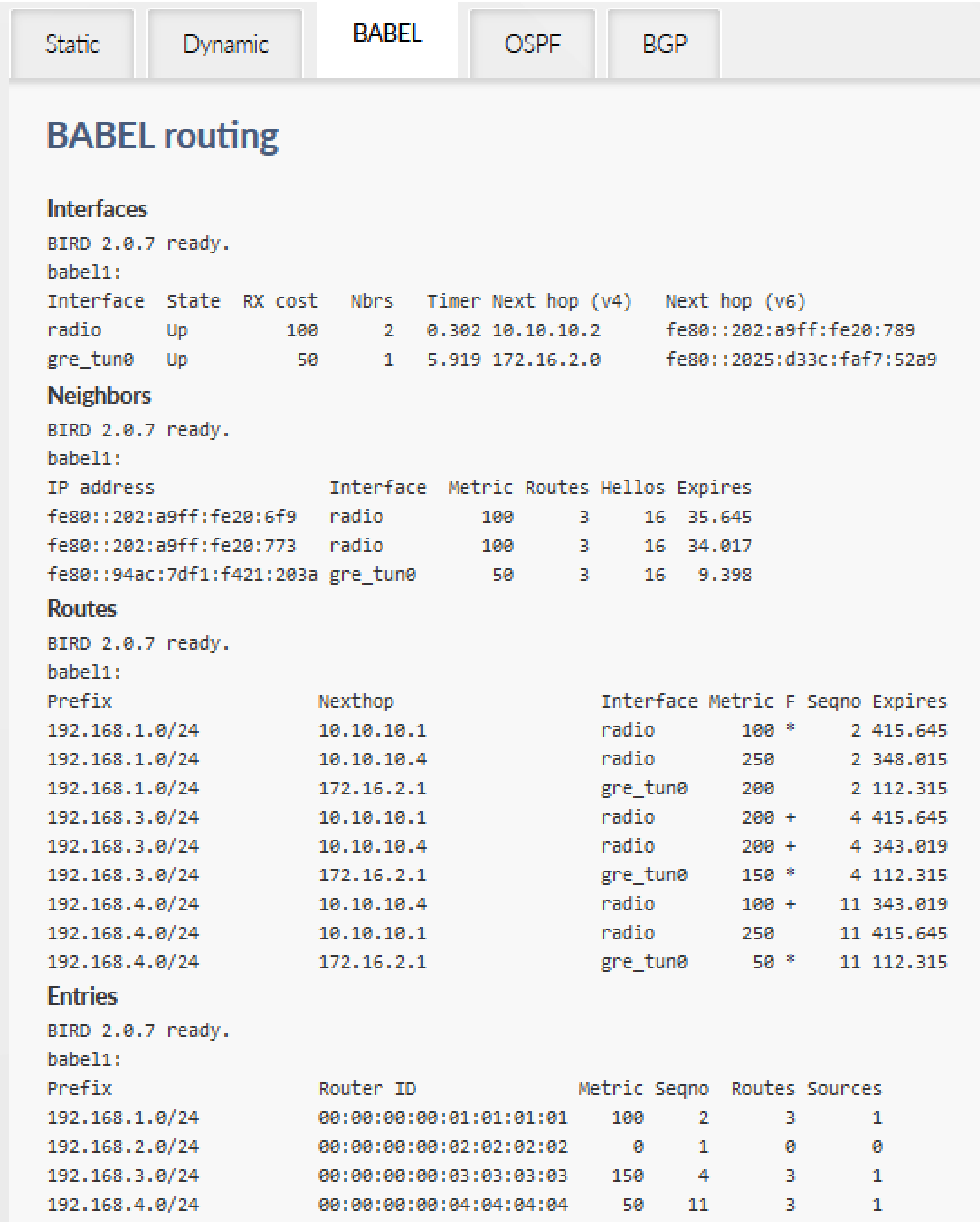


Fig. 5.11: RipEX_B – Pings over cellular/LTE

If these pings are not working correctly, check your APN setup and GRE tunnel configuration.

Go to the Diagnostics – Routing menu and check the Babel statistics.



Static | Dynamic | BABEL | OSPF | BGP

BABEL routing

Interfaces

```
BIRD 2.0.7 ready.
babel1:
Interface  State  RX cost   Nbrs   Timer Next hop (v4)    Next hop (v6)
radio      Up         100      2   0.302 10.10.10.2       fe80::202:a9ff:fe20:789
gre_tun0   Up          50      1   5.919 172.16.2.0       fe80::2025:d33c:faf7:52a9
```

Neighbors

```
BIRD 2.0.7 ready.
babel1:
IP address                Interface  Metric Routes Hellos Expires
fe80::202:a9ff:fe20:6f9   radio         100      3     16  35.645
fe80::202:a9ff:fe20:773   radio         100      3     16  34.017
fe80::94ac:7df1:f421:203a gre_tun0       50      3     16   9.398
```

Routes

```
BIRD 2.0.7 ready.
babel1:
Prefix                    Nexthop                    Interface Metric F Seqno Expires
192.168.1.0/24            10.10.10.1                 radio        100 *     2 415.645
192.168.1.0/24            10.10.10.4                 radio        250       2 348.015
192.168.1.0/24            172.16.2.1                 gre_tun0     200       2 112.315
192.168.3.0/24            10.10.10.1                 radio        200 +     4 415.645
192.168.3.0/24            10.10.10.4                 radio        200 +     4 343.019
192.168.3.0/24            172.16.2.1                 gre_tun0     150 *     4 112.315
192.168.4.0/24            10.10.10.4                 radio        100 +    11 343.019
192.168.4.0/24            10.10.10.1                 radio        250      11 415.645
192.168.4.0/24            172.16.2.1                 gre_tun0      50 *    11 112.315
```

Entries

```
BIRD 2.0.7 ready.
babel1:
Prefix                    Router ID                Metric Seqno  Routes Sources
192.168.1.0/24            00:00:00:00:01:01:01:01   100      2       3       1
192.168.2.0/24            00:00:00:00:02:02:02:02     0      1       0       0
192.168.3.0/24            00:00:00:00:03:03:03:03   150      4       3       1
192.168.4.0/24            00:00:00:00:04:04:04:04    50     11       3       1
```

Fig. 5.12: RipEX_B – Babel Routing

There are already 3 routes to each destination in RipEX_B radio. Two are using RF channel and one the GRE tunnel. Cost/Metric via LTE is set to “50” so you can see a route to 192.168.4.0/24 via “gre_tun0” with a Metric equal to 50.

You can test the scenario by unplugging the cellular antennas or turning off RipEX_B completely. Keep the ICMP ping running from your laptop to 192.168.4.1 IP address again.

Check Statistics and Monitoring menus. It is NOT possible to monitor cellular interface in RipEX2 firmware 2.0.5.0 and older. Also note that Monitoring via Remote access is only available since RipEX2 firmware 2.0.5.0.

Neighbors

```
BIRD 2.0.7 ready.
babel1:
IP address                   Interface  Metric Routes Hellos Expires
fe80::202:a9ff:fe20:6f9      radio         100      3     16  40.940
fe80::202:a9ff:fe20:773      radio         114      3     16  39.314
fe80::94ac:7df1:f421:203a gre_tun0     65535      0      3   4.534
```

Fig. 5.13: RipEX_A – neighbors (cellular antenna removed from RipEX_B)

You can also check if the cellular interface is “up” or “down” in the SETTINGS – Interfaces – Cellular menu.

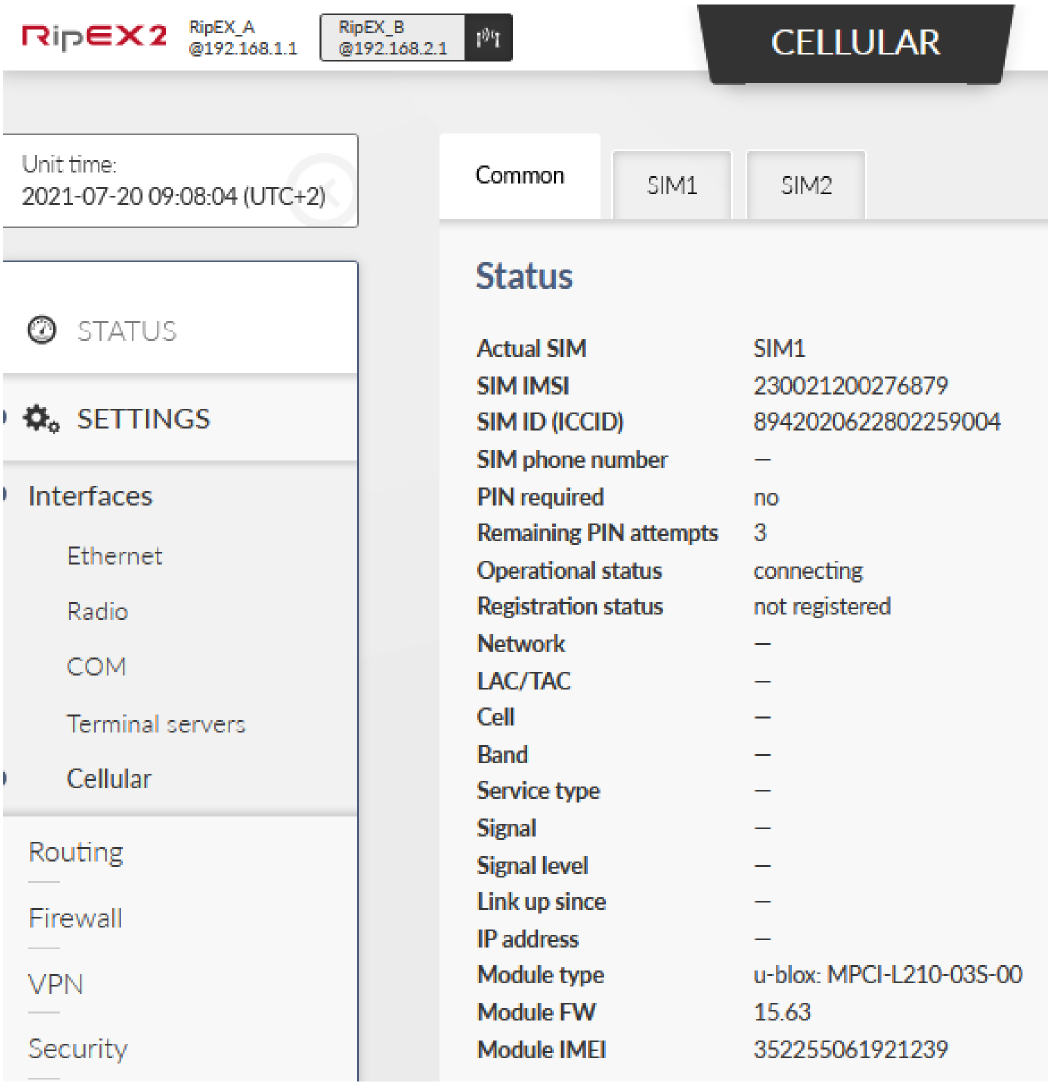


Fig. 5.14: RipEX_B – Cellular/LTE connection down

6. Basic Babel and OSPF combination

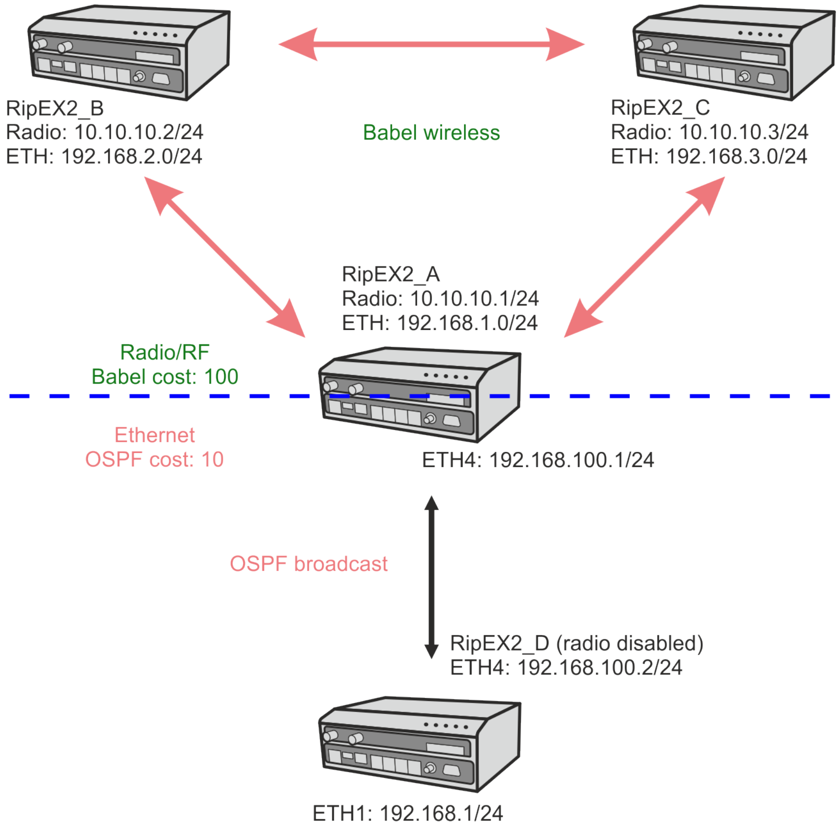


Fig. 6.1: Example 6 – Babel and OSPF diagram

6.1. Description

The sixth example shows a network combining Babel and OSPF. OSPF is configured in RipEX_A and RipEX_D units, whereas Babel is configured in RipEX_A, B and C. These two protocols are neighboring in one Autonomous System Border Router “ASBR” (RipEX_A) which divides the whole network into two parts. Bandwidth optimized Babel on the Radio segment and standardized, widely-used and fast OSPF on the Ethernet segment.

These two parts are interconnected together utilizing default gateways. E.g., Radio network is completely routed by Babel and all other routes are routed to ETH4 interface of RipEX_A via a default gateway rule (0.0.0.0/0). In other words, there is no other export of routing rules.

Without dynamic protocols, all routes would need to be filled manually and statically.

Startup configurations are taken from first example. See the required changes below. It is required to change RipEX_A and RipEX_D settings only.

6.2. RipEX_A Configuration

Add a 2nd IP/Mask to the “bridge” Ethernet interface. A new 192.168.100.1/24 IP address will be used for Ethernet connection to RipEX_D and OSPF. You could also detach a particular ETH port and assign this IP to this designated port.

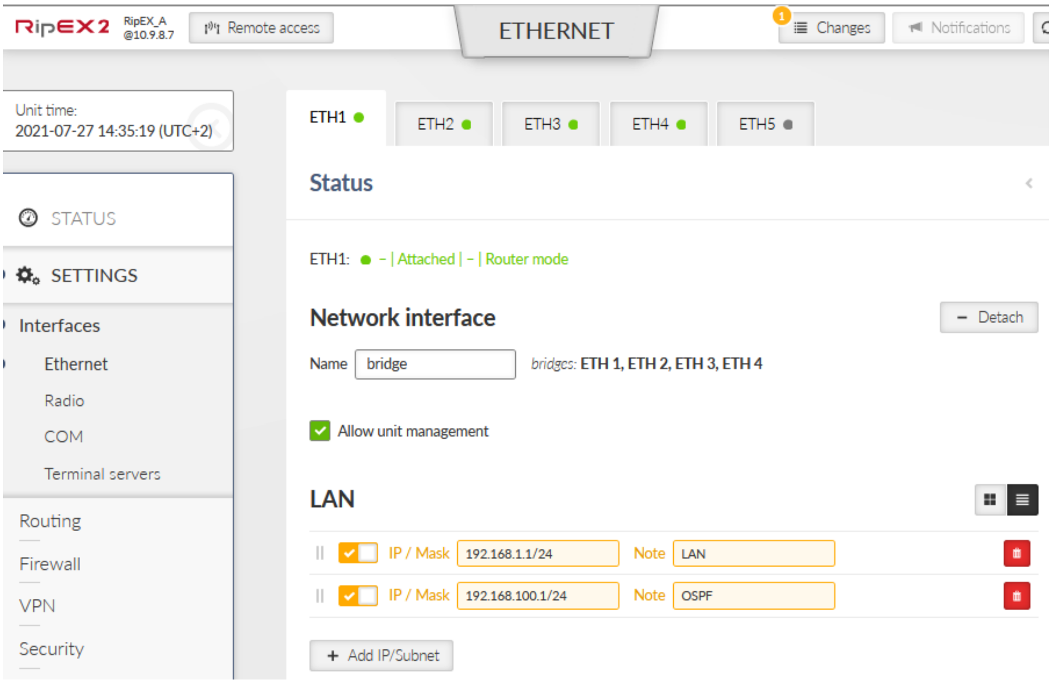


Fig. 6.2: RipEX_A – Ethernet bridge

Add a new “default route” to current Babel Static rules. This default route will be used between OSPF and Babel network segments.

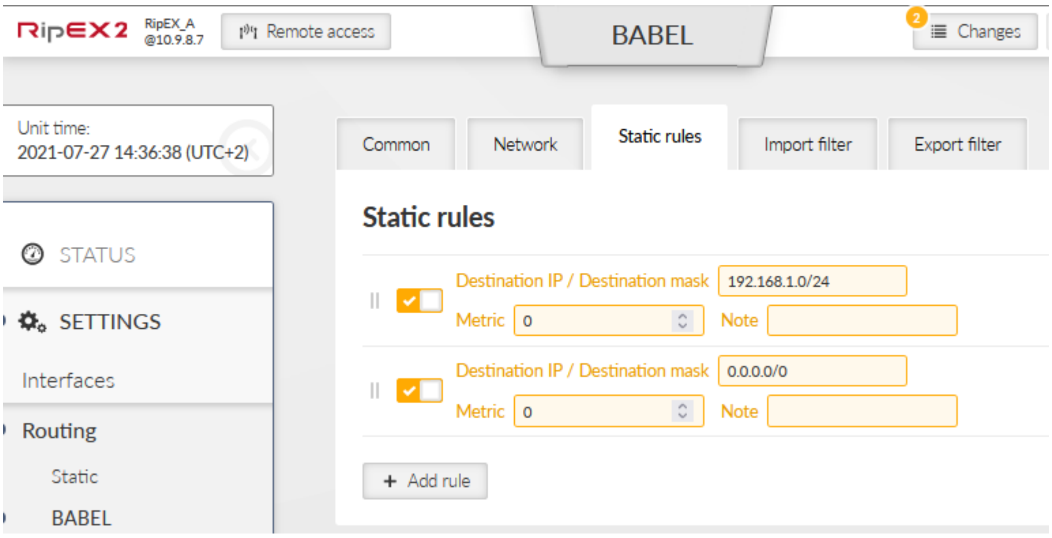


Fig. 6.3: RipEX_A – Babel Static rules (def. gateway)

Go to the OSPF Routing menu and enable OSPF dynamic protocol. The Router ID is shared among all dynamic protocols.

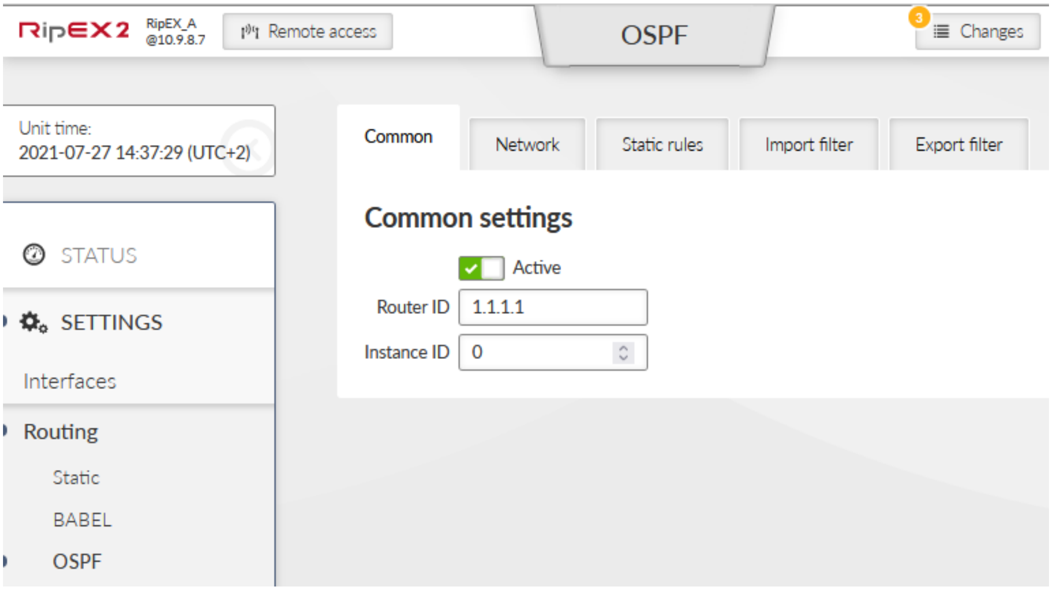


Fig. 6.4: RipEX_A – OSPF activation

Create a new OSPF Area ID 0.0.0.0 (so called “backbone area” which has to be used in OSPF and all other areas must be directly connected to it).

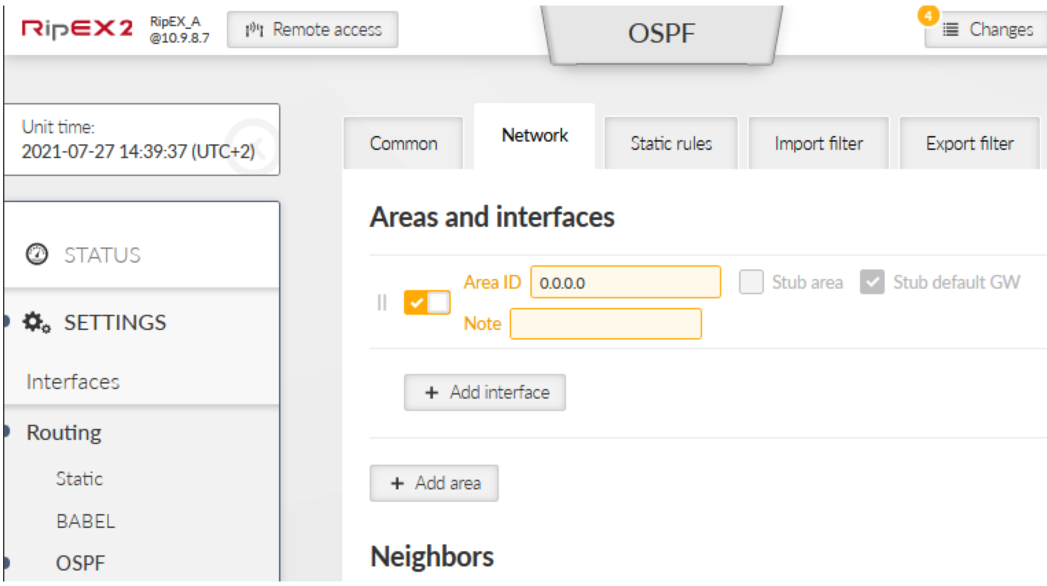


Fig. 6.5: RipEX_A – New OSPF backbone Area

Add a new interface under this backbone Area. Change the following parameters:

Interface	if_bridge
Network IP/Mask	192.168.100.0/24
Network type	Broadcast

Multiple OSPF routers on a link on which all participants can hear each other. The link allows both broadcast and multicast for OSPF mechanism. All participants vote for one Designated Router (DR) and one Backup Designated Router (BDR) which are responsible for resending routing updates among other routers (to limit protocol overhead data).

Hello interval 2

Ethernet is a fast link so the interval can be low (i.e., fast OSPF processes).

Other parameters stay in default values.

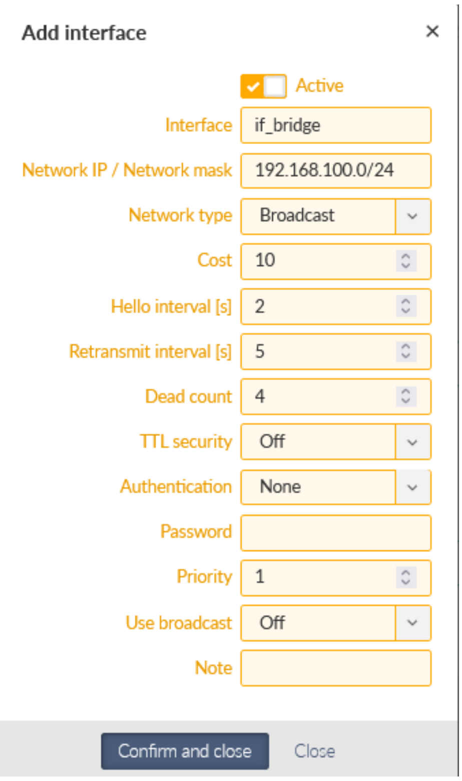


Fig. 6.6: RipEX_A – New OSPF interface

Go to the Static rules tab and add two rules.

- 192.168.1.0/24
  - Metric type 2
  - Metric 1000
- 0.0.0.0/0
  - Metric type 2
  - Metric 1000

It is not important if the metric type is “1” or “2” in this example, but it is used to distinguish rules which came to OSPF from Babel (and cannot be exported back to Babel).

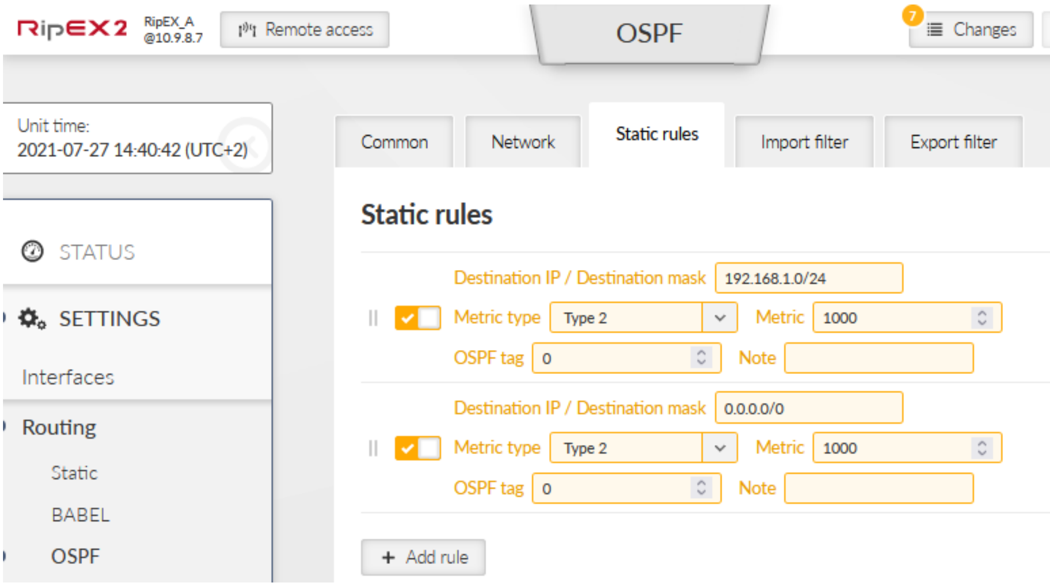


Fig. 6.7: RipEX_A – OSPF static rules

Go to the last required tab and add an Import filter with the following settings. Only the “Local preferred source address” is to be filled – with local ETH IP 192.168.1.1 so that packets generated in this RipEX2 unit use 192.168.1.1 as their Source IP.

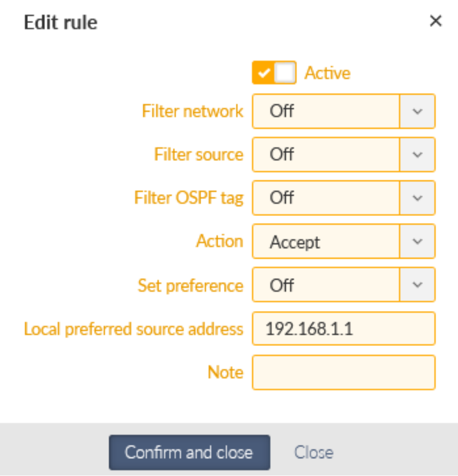


Fig. 6.8: RipEX_A – OSPF Import filter

There is no change required in **RipEX_B** and **RipEX_C** units.

6.3. RipEX_D Configuration

Add a 2nd IP/Mask to the “bridge” Ethernet interface. A new 192.168.100.2/24 IP address will be used for Ethernet connection to RipEX_A and OSPF. You could also detach a particular ETH port and assign this IP to this designated port.

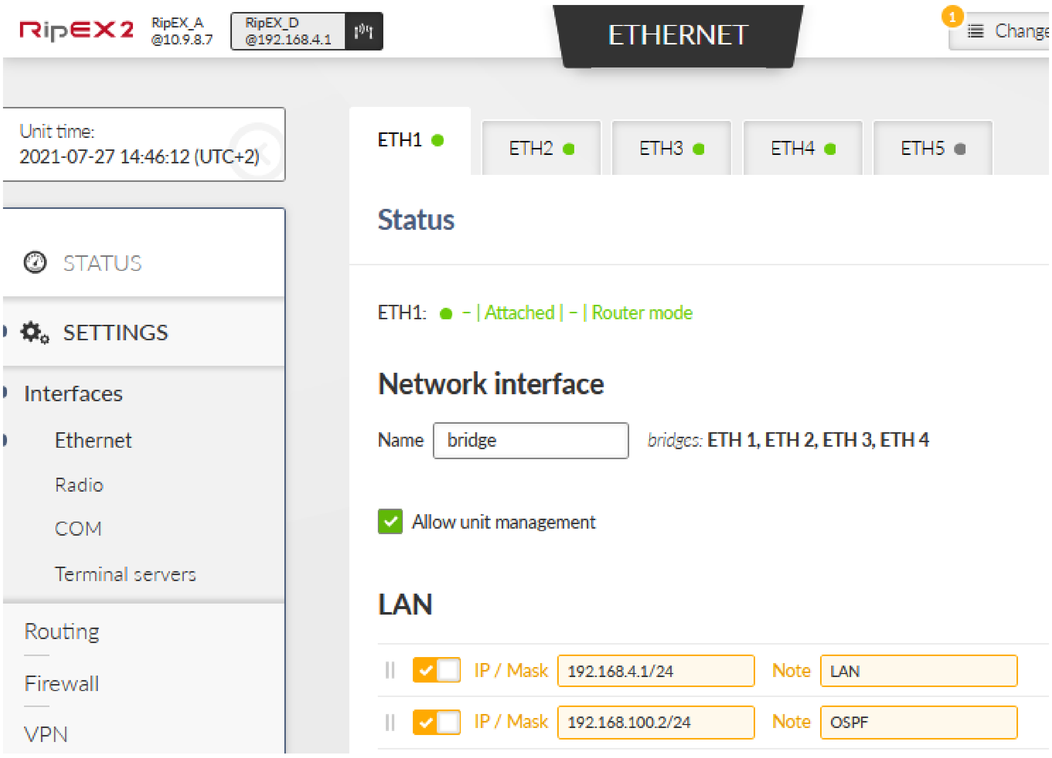


Fig. 6.9: RipEX_D – Ethernet configuration

Disable the Radio protocol so there is no Radio/RF communication, only Ethernet connection to RipEX_A.

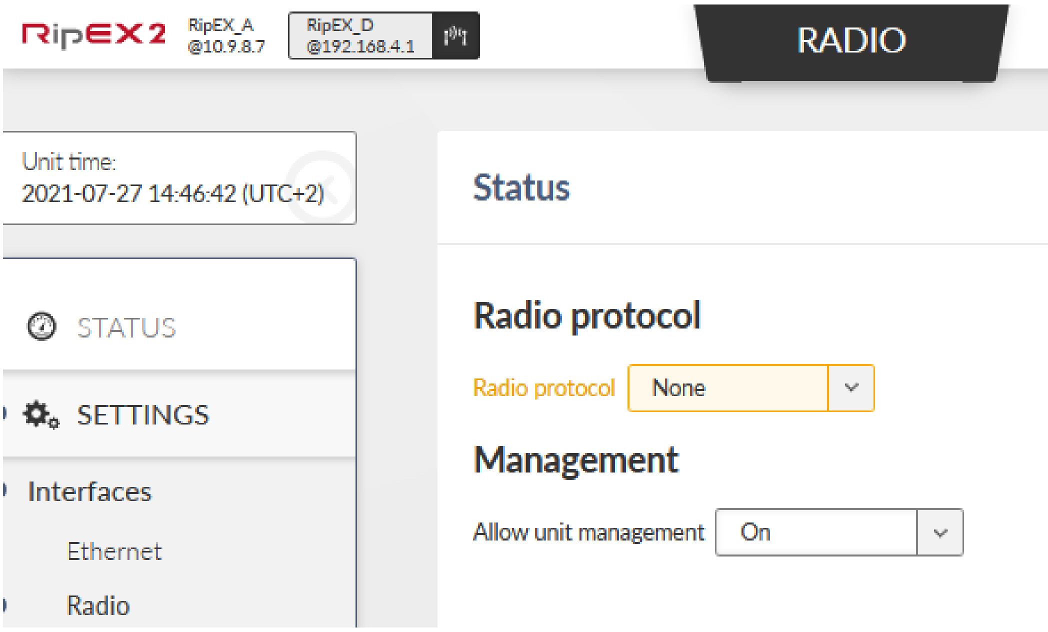


Fig. 6.10: RipEX_D – Radio communication disabled

Disable Babel protocol.

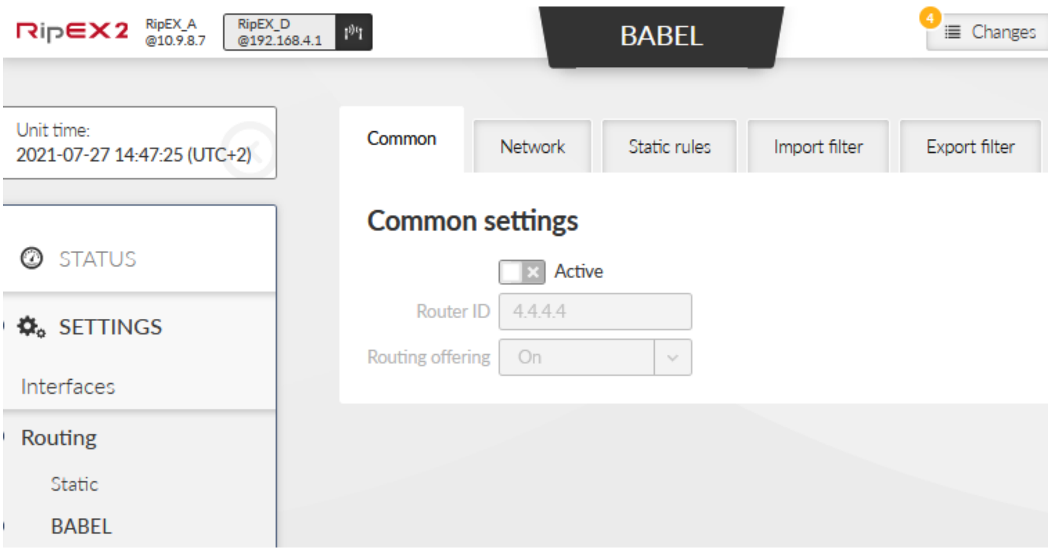


Fig. 6.11: RipEX_D – Babel protocol disabled

Enable OSPF with the same Router ID 4.4.4.4.

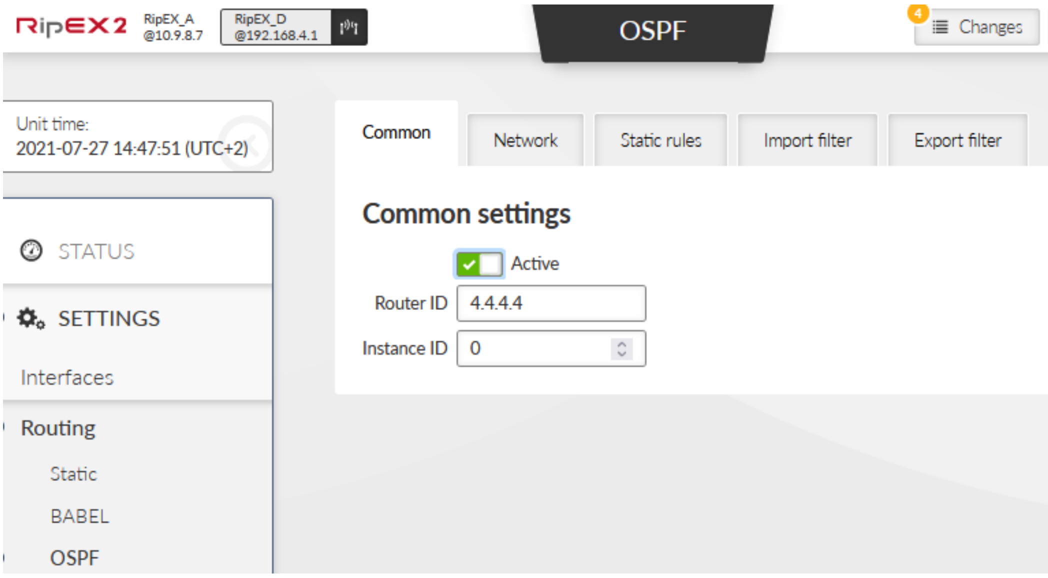


Fig. 6.12: RipEX_D – OSPF protocol enabled

Add one OSPF interface within a new backbone Area ID 0.0.0.0.

Interface	if_bridge
Network IP/Mask	192.168.100.0/24
Network type	Broadcast
Hello interval	2

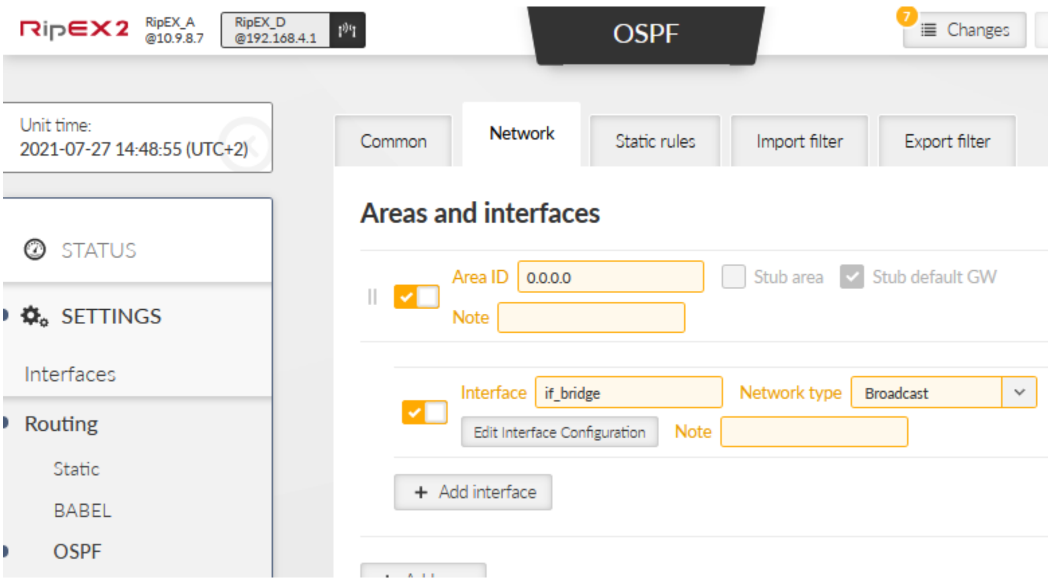


Fig. 6.13: RipEX_D – New OSPF Area and Interface

Details:

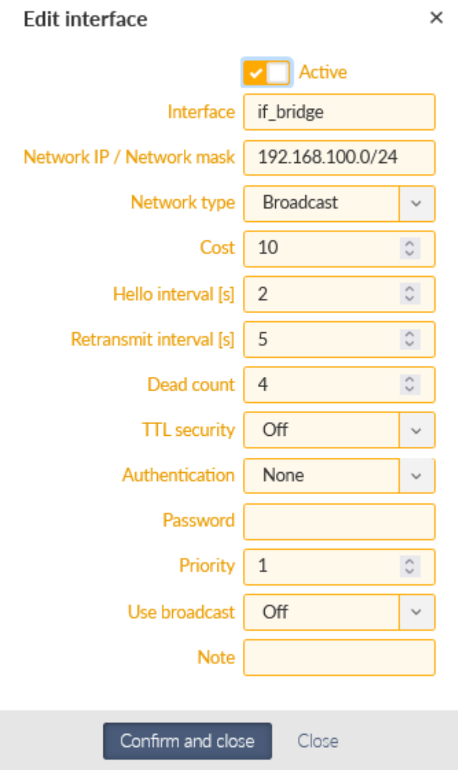


Fig. 6.14: RipEX_D – OSPF interface details

Add one Static rule so that local 192.168.4.0/24 is propagated through OSPF protocol. Set the same Metric type (2) and Metric (1000) as in RipEX_A.

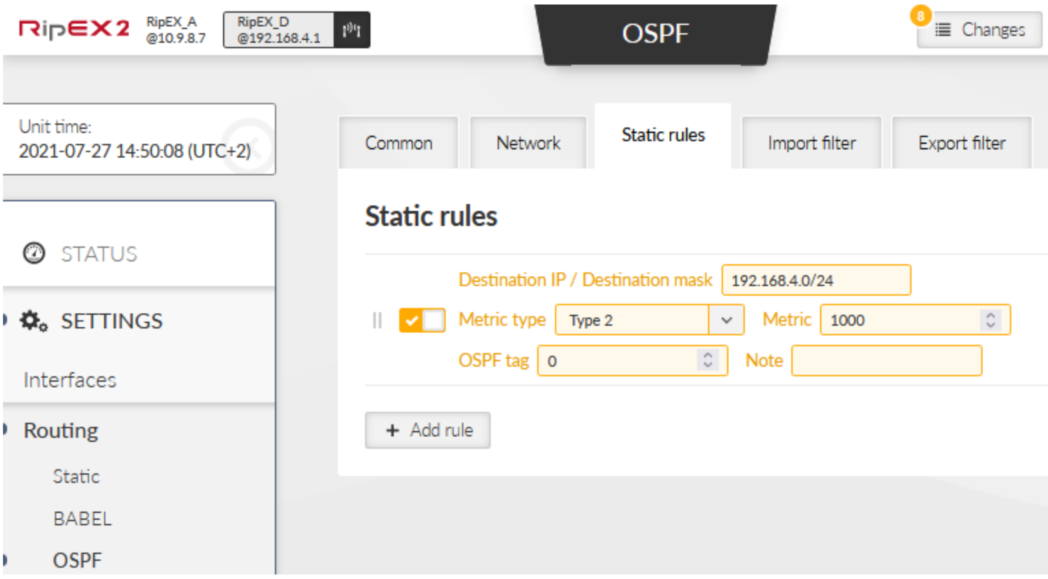


Fig. 6.15: RipEX_D – OSPF Static rules

Finally, add one Import filter with Local preferred source address equal to 192.168.4.1.

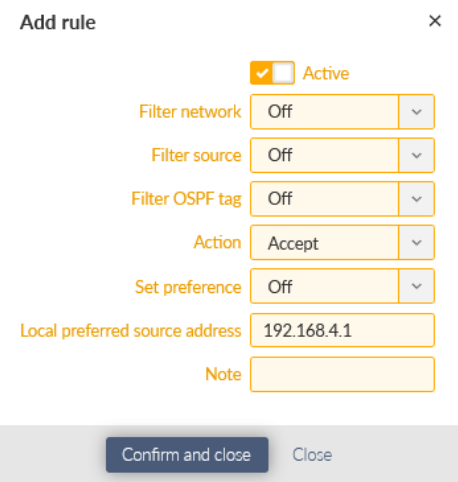


Fig. 6.16: RipEX_D – OSPF Import filter

6.4. Diagnostics and Testing

You can attach your laptops to RipEX2 units and do some PING tests. You can also configure some SCADA protocol traffic simulation.

The “issue” with testing this scenario on the desk is that each radio can hear each other (except RipEX_D). Thus, it is not possible to force Babel and Radio channel to send traffic via a repeater by default and only if this fails, sending data directly. The traffic always uses a direct connection. Only if you had some attenuators or you use it in the field, there should be situations in which traffic goes via two radio hops.

Also, the OSPF and Babel are neighboring in one ASBR router (RipEX_A) – so if you turn off RipEX_A, then RipEX_D loses its connection to the radio segment (and vice versa). There is no other way.

So, check the correct routing is set and used in all routers and also check the ping works among all units. Typical outputs you should see are:

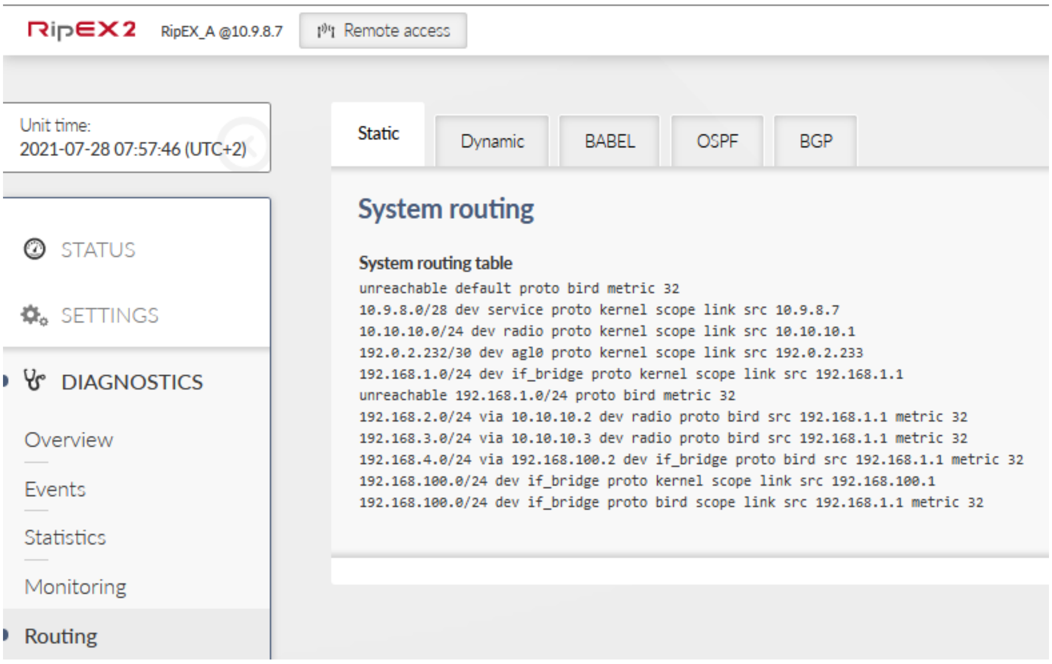


Fig. 6.17: RipEX_A – System routing

RipEX_A should have /24 routes to all LANs behind every other RipEX2 in this example.

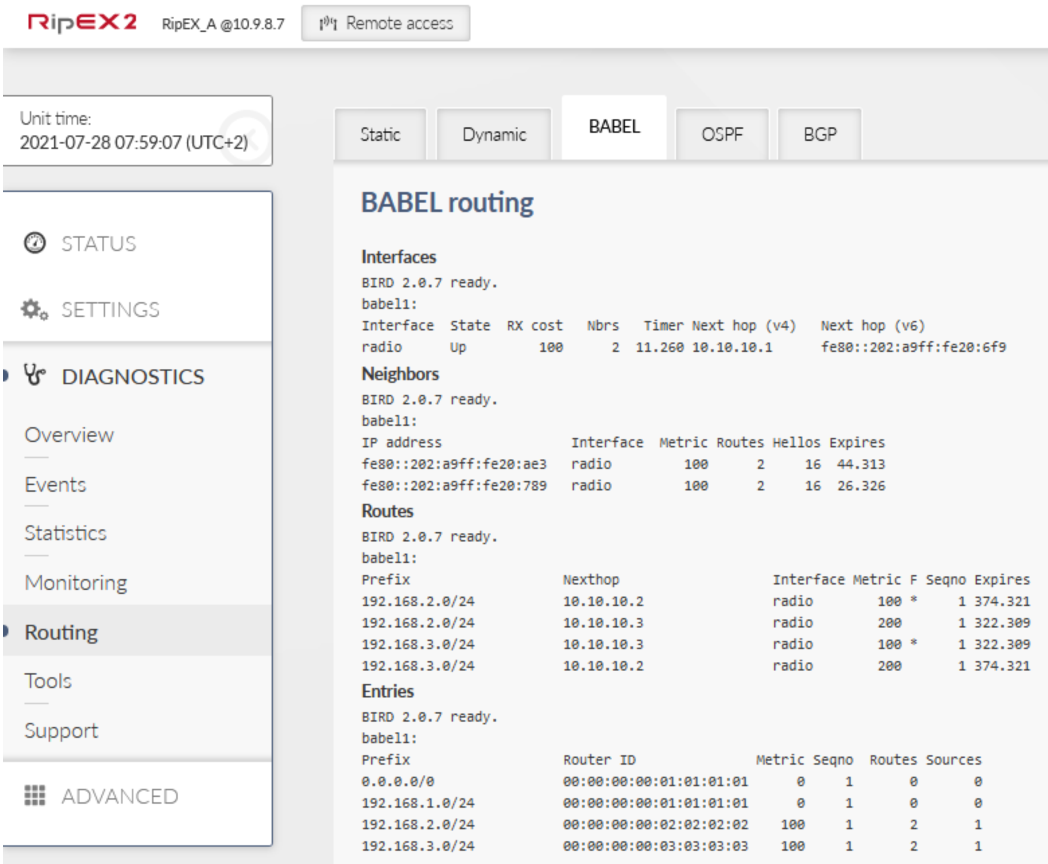


Fig. 6.18: RipEX_A – Babel routing

In the Babel output, you should see two neighbors and a direct metric to each with a cost of 100.

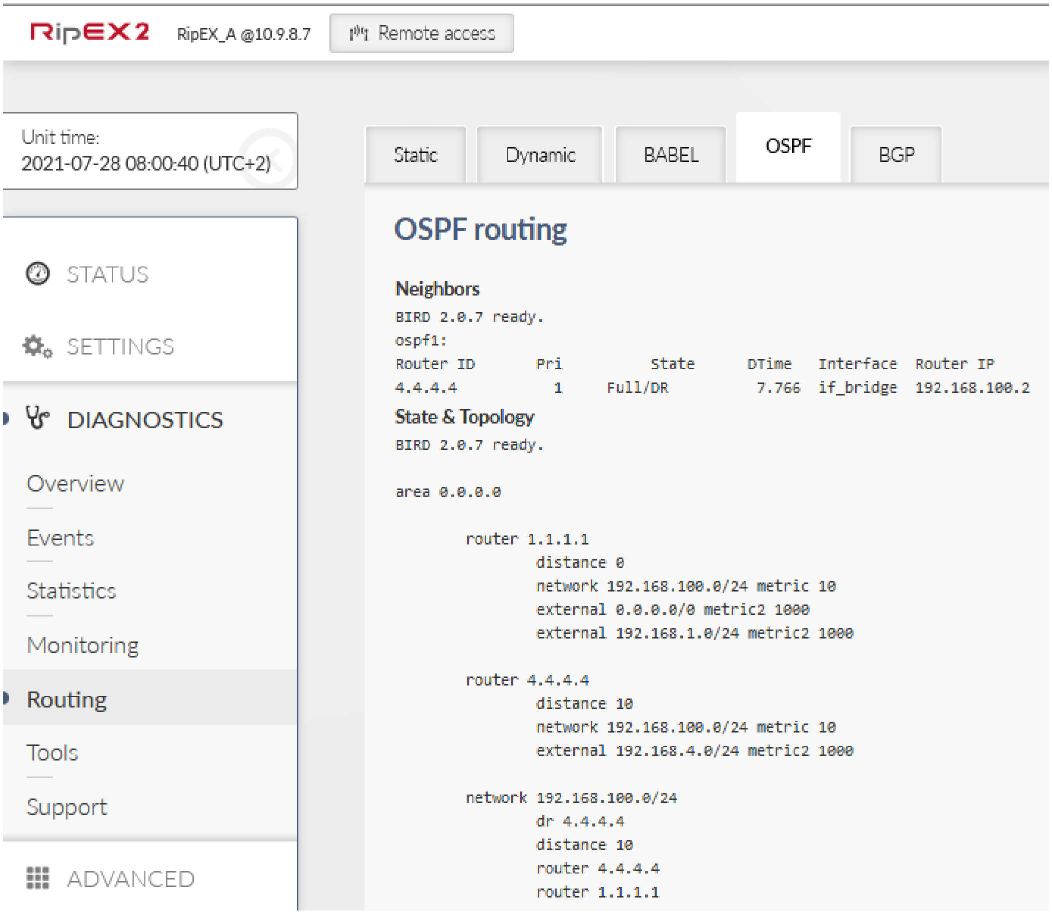


Fig. 6.19: RipEX_A – OSPF output

There should be one 4.4.4.4 neighbor with state either Full/DR or Full/BDR.

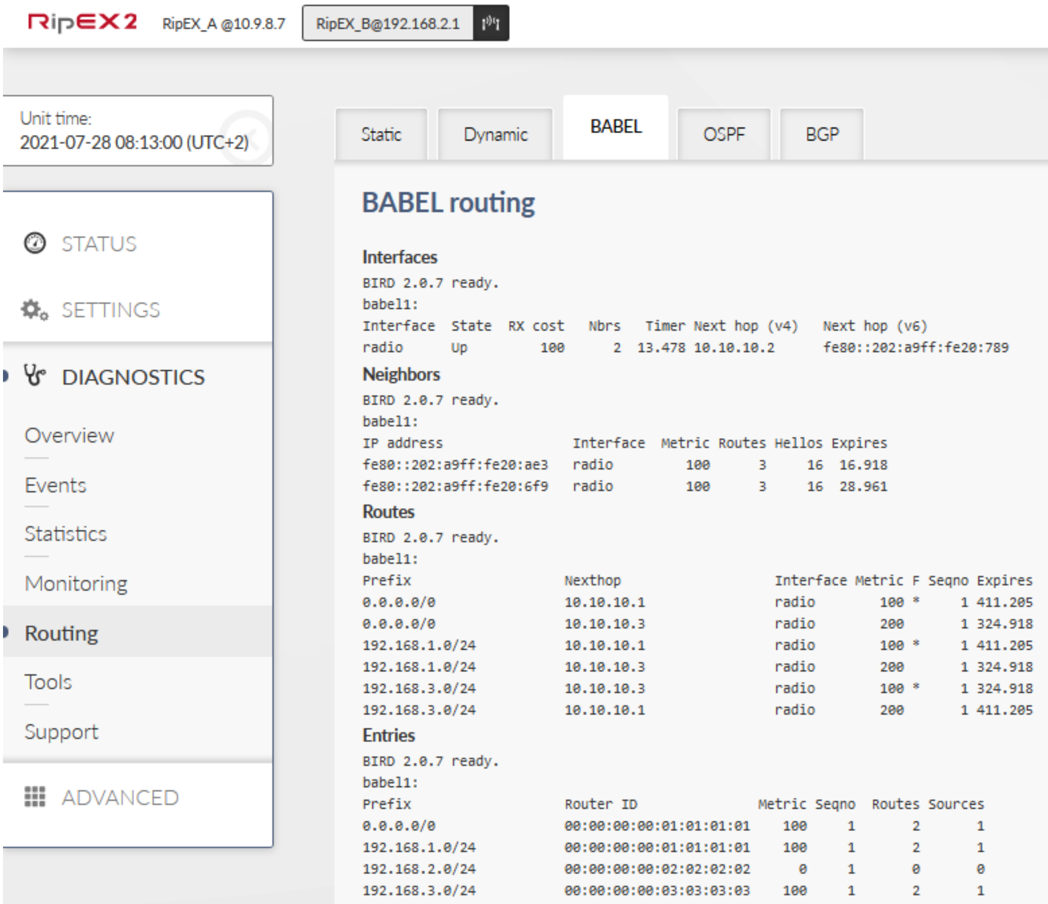


Fig. 6.20: RipEX_B – Babel routing

RipEX_B should have routes to its Radio neighbors (192.168.1.0/24 and 192.168.3.0/24) with Metric equal to 100 and one default gateway rule back to RipEX_A (also with a Metric equal to 100). The same is for RipEX_C.

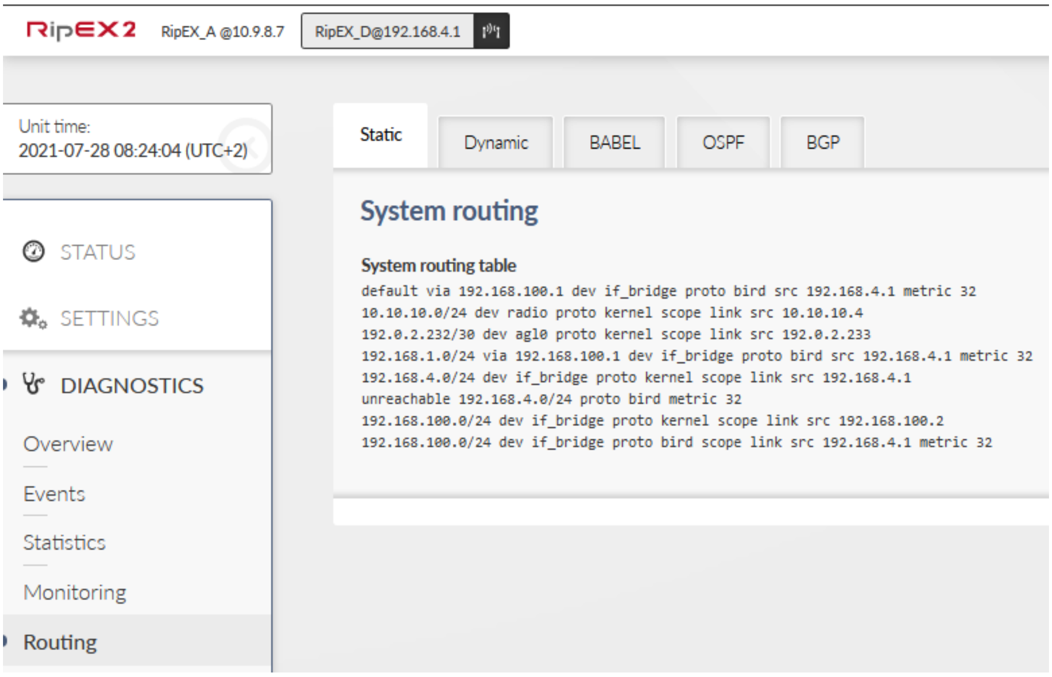


Fig. 6.21: RipEX_D – System routing

RipEX_D should have a /24 link to 192.168.1.0 subnet via its if_bridge interface and gateway 192.168.100.1 (RipEX_A) and a default gateway (also 192.168.100.1) for the whole radio segment.

All the routes are obtained via OSPF. There is no Babel configured.

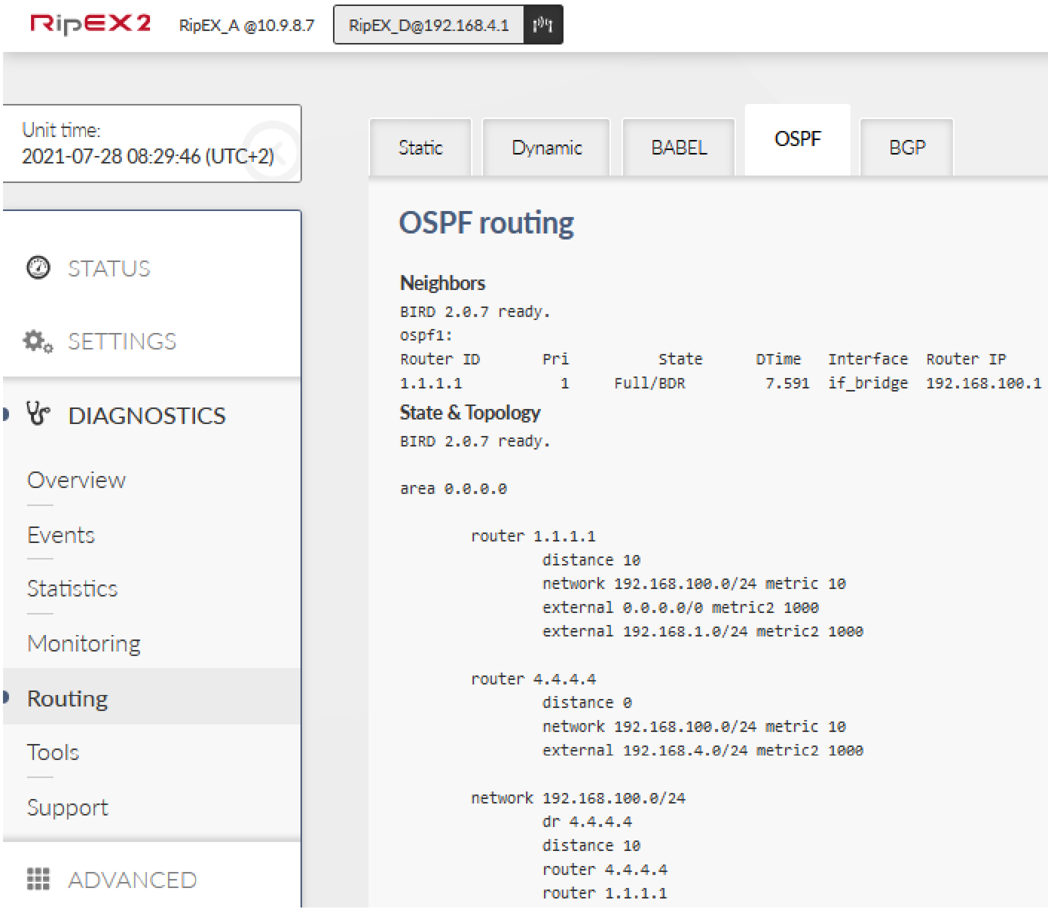


Fig. 6.22: RipEX_D – OSPF routing

7. Advanced Babel and OSPF combination

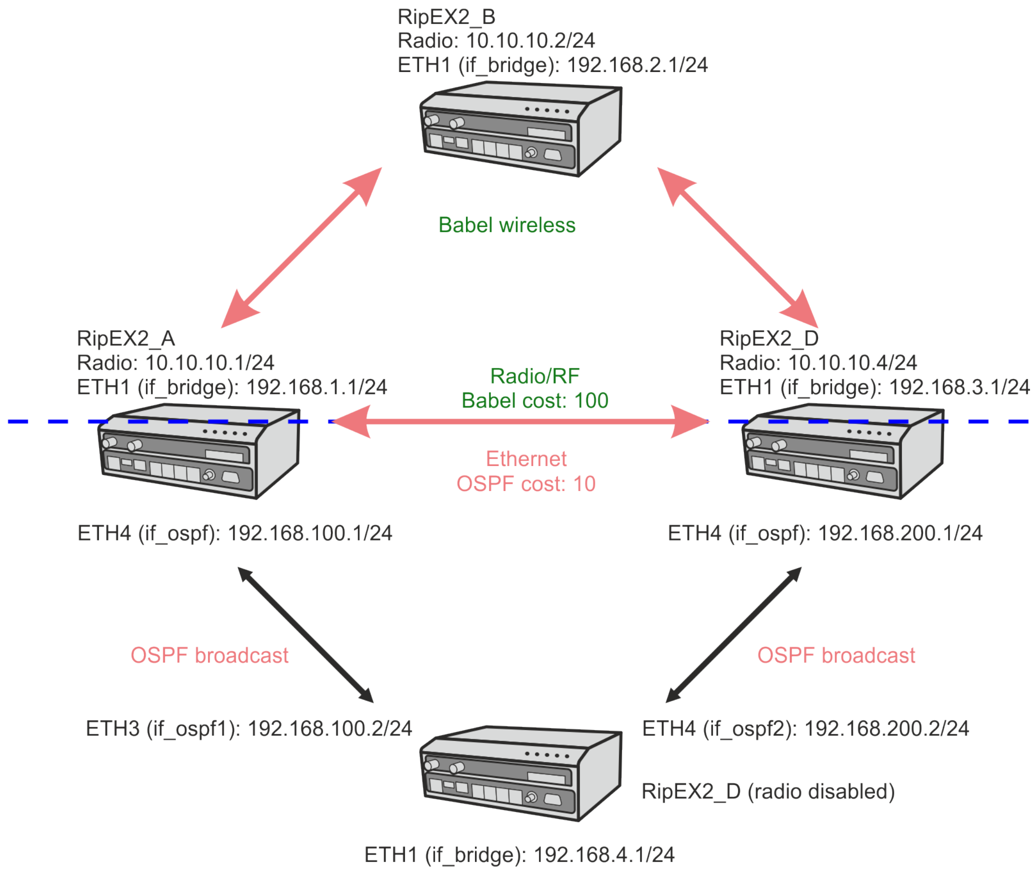


Fig. 7.1: Example 7 Babel and OSPF diagram 3

7.1. Description

The seventh example extends the sixth one to show even more from OSPF and Babel combination. There are two ASBR routers between Babel and OSPF network segments. By default, the primary path should go over RipEX_A and a backup path is over RipEX_C.

We set the static preferences of routes to be propagated so that RipEX_A is a preferred ASBR. Routing rules are being exchanged between mentioned segments, not just default gateways.

This example continues from the sixth example and only changes are explained below.

7.2. RipEX_A Configuration

First of all, let's divide one common bridge interface on all Ethernet ports. This is not mandatory, but shows a different approach compared to the previous example, i.e., one IP address/subnet on each individual ETH port.

To do so, go to the SETTINGS – Interfaces – Ethernet menu. Delete the 2nd subnet from the primary bridge interface.

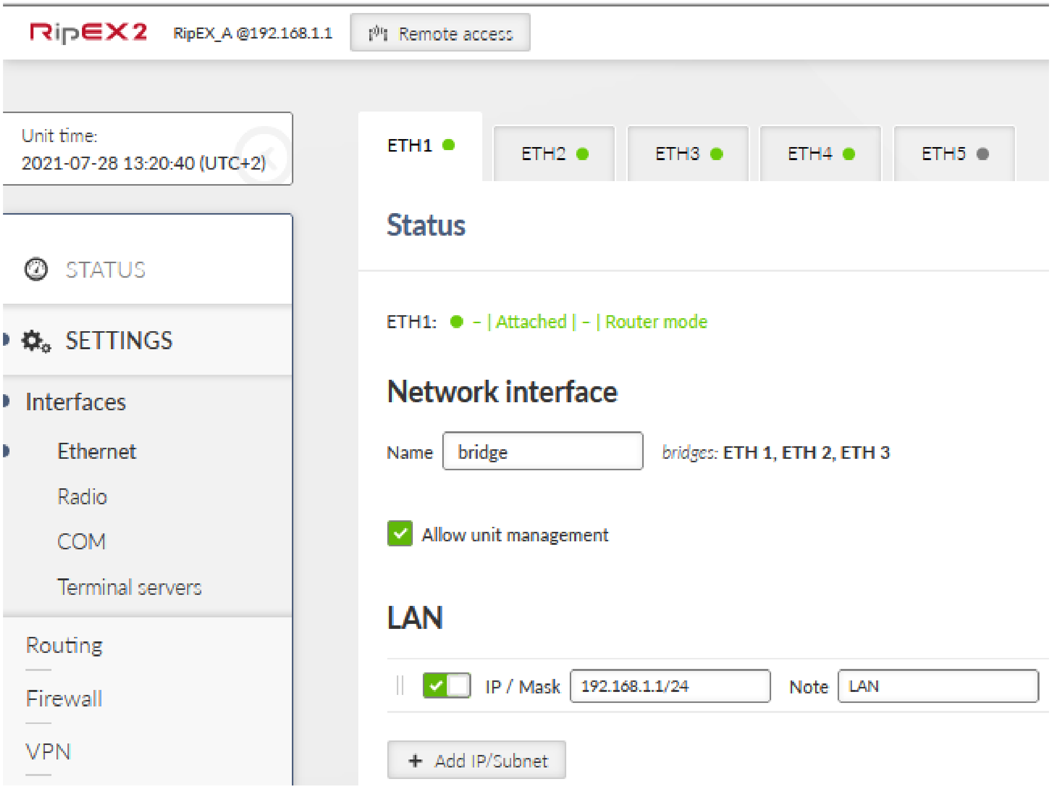


Fig. 7.2: RipEX_A – ETH1 configuration

Go to the ETH4 tab. Detach it from the current bridge and add a new interface called “ospf”. Add IP/subnet equal to 192.168.100.1/24.

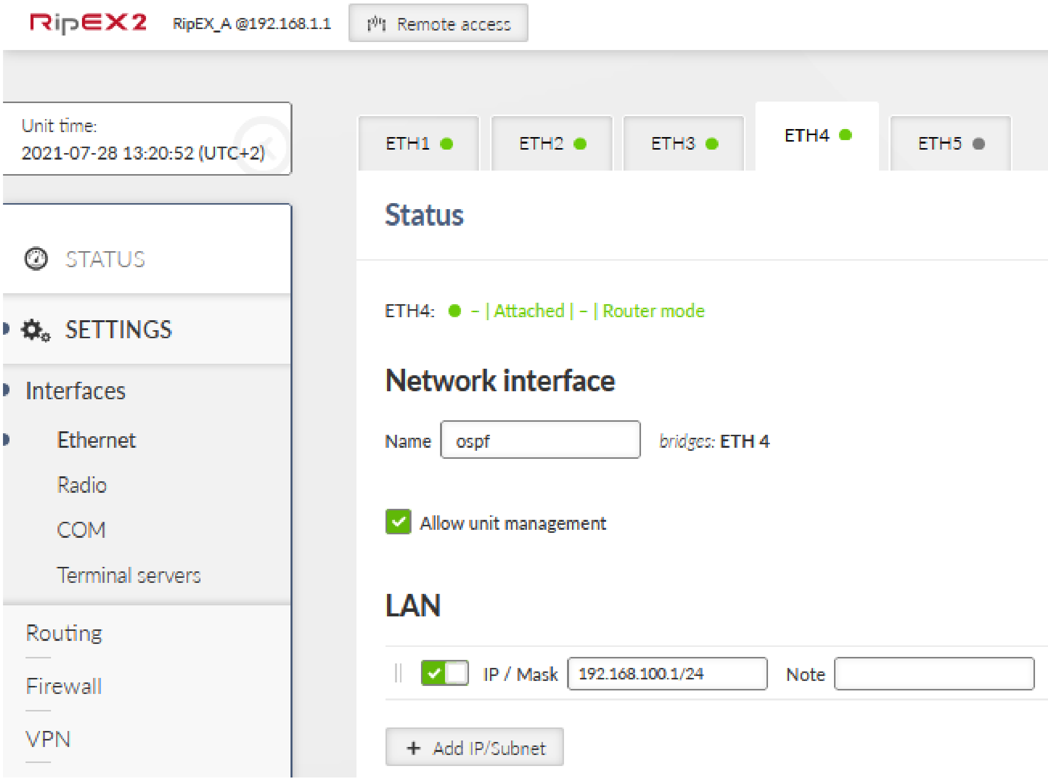


Fig. 7.3: RipEX_A – ETH4 configuration

We need to change OSPF setup now. Go to the Routing – OSPF menu and select a Network tab. Change the Interface name to “if_ospf” to reflect ETH4 change and a new interface name.

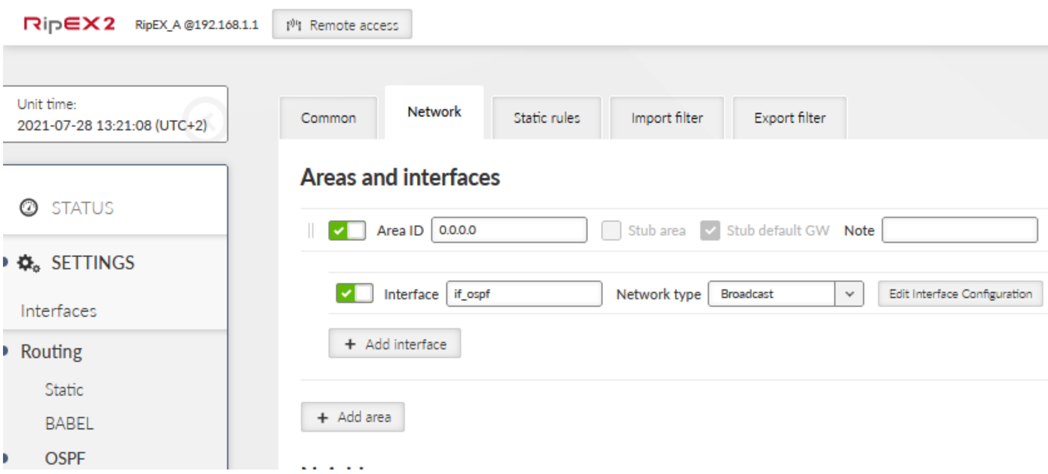


Fig. 7.4: RipEX_A – OSPF Network interface

Change the tab to “Static rules”. We only keep one rule 192.168.1.0/24. Delete the default route rule, because particular routes will be propagated, not a default gateway.

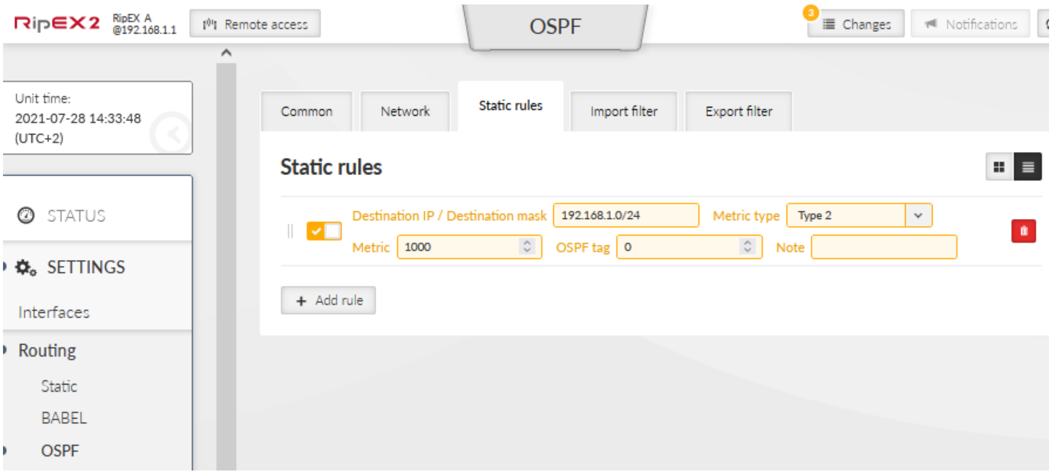


Fig. 7.5: RipEX_A – OSPF Static rules

Go to the “Import filter” tab and create two rules. The first one is to import external routes (EXT2) with preference 150 (lower/worse than Babel which has 210 by default). Set the Local preferred source address to local ETH1 IP 192.168.1.1.

These are the rules which were imported from Babel to OSPF in a different ASBR. We could also use OSPF tag filter instead of source filter.

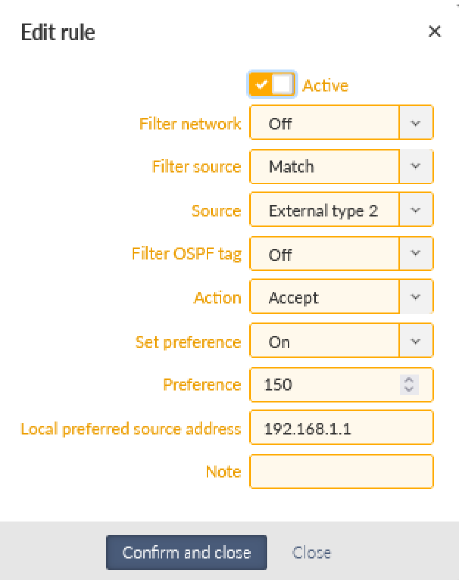


Fig. 7.6: RipEX_A – Import filter, 1st rule

Import all other rules and set a preference to 250 (higher/better than Babel). Make sure the order of rules is met, **the order is important**!

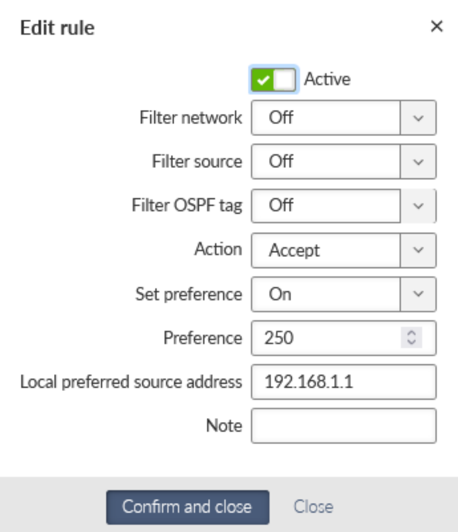


Fig. 7.7: RipEX_A – Import filter, 2nd rule

We also need to set Export filter. OSPF export filter rules define set of routing rules to be exported from the unit into the OSPF area.

Due to this rule, OSPF export rules from Babel with Metric type 2 (EXT2) and Metric equal to 1000 (RipEX_A is a preferred ASBR/router).

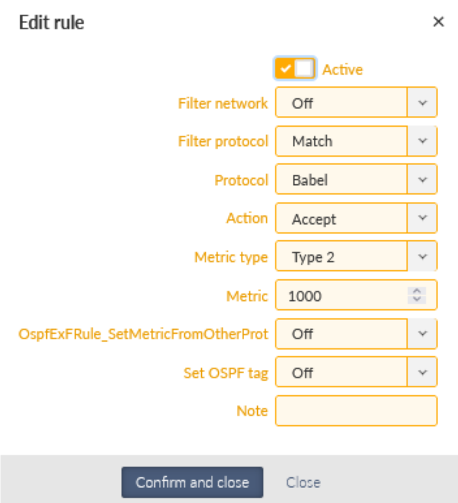


Fig. 7.8: RipEX_A – Export filter

Now, select the Routing – Babel menu. First of all, keep only one Static rule (192.168.1.0/24, metric 0).

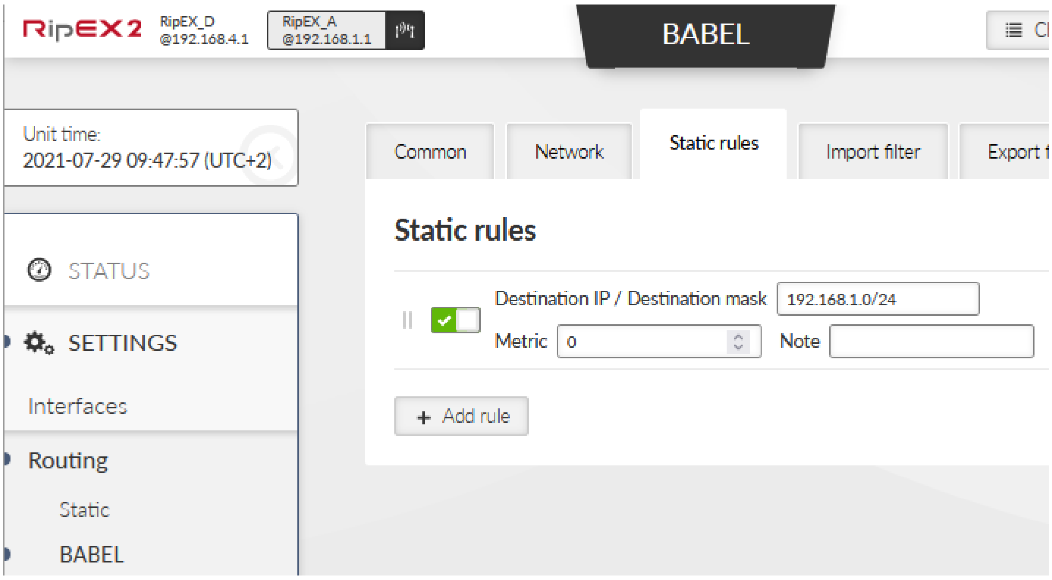


Fig. 7.9: RipEX_A – Babel Static rules

The last step in RipEX_A setup is to create a new Babel Export filter rule.

This rule exports rules from OSPF with EXT1 type and sets a metric to 1000 (preferred ASBR/router, higher number than any sum of metrics in Babel network/segment).

Note

EXT2 rules mark rules originally from Babel network and they are not exported back. EXT1 rules mark rules from OSPF network which are forwarded to Babel. It is also possible to use OSPF tag filter – marking rules from Babel which are exported to OSPF with this tag.

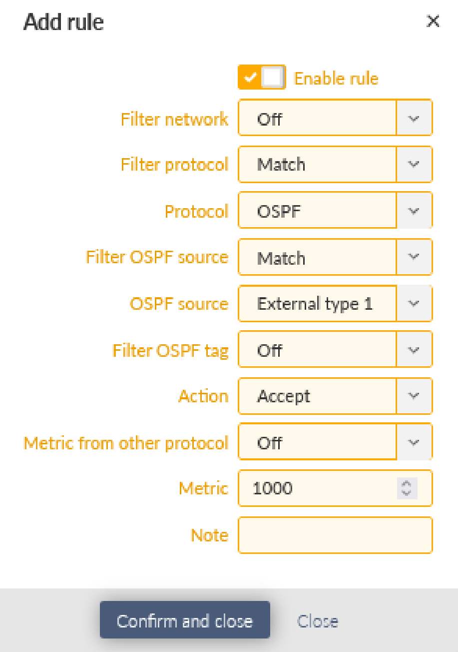


Fig. 7.10: RipEX_A – Babel export rule

Save all the changes. RipEX_B does not require any change, continue with RipEX_C. If the remote access is not possible currently, we recommend using the management USB/ETH access.

7.3. RipEX_C Configuration

RipEX_C is a 2nd ASBR (router) on the OSPF/Babel border and is used as a backup OSPF router. Most of the required configuration steps are the same as in RipEX_A, but we make sure to set various metrics/preferences worse (so RipEX_A is a primary ASBR).

Start with detaching ETH4 port and setting its new name “ospf” and IP/Mask equal to 192.168.200.1/24. Do not forget to delete this IP from ETH1 settings.

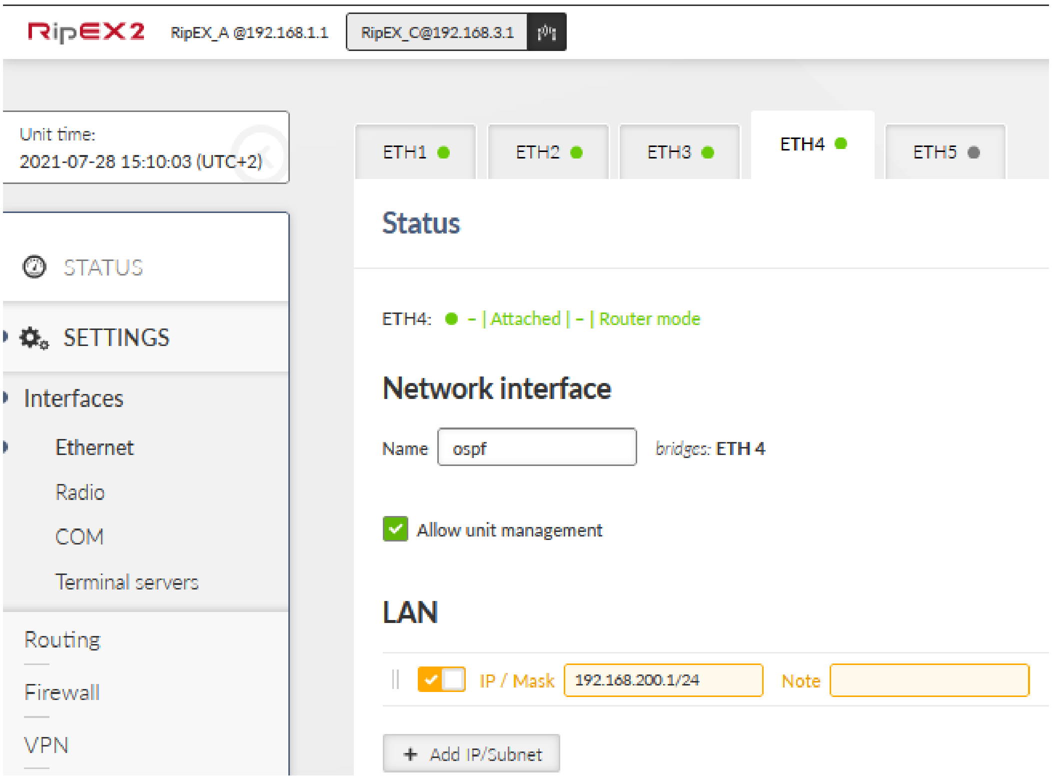


Fig. 7.11: RipEX_C – ETH4 configuration

Go to the Routing – OSPF menu and activate its functionality. The Router ID is shared among all dynamic protocols so keep it 3.3.3.3.

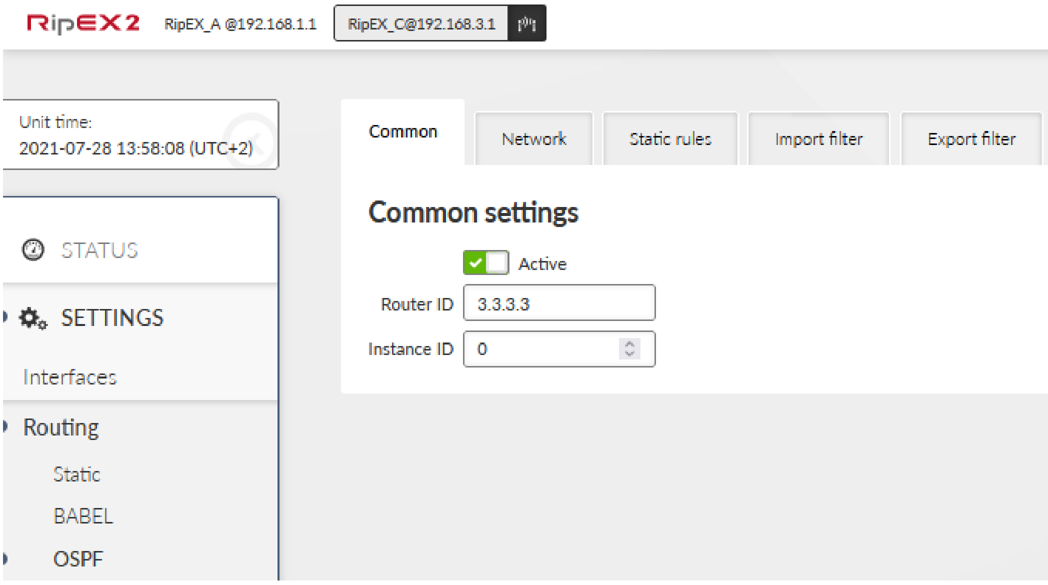


Fig. 7.12: RipEX_C – OSPF activation

Go to next tab “Network” and create a new backbone area 0.0.0.0 with the following new interface. The interface must be named per the ETH4 name, so “if_ospf”. Set the correct network on this port 192.168.200.0/24 and decrease the Hello interval to 2 seconds, because Ethernet is a fast interface. Other parameters can stay in default values (e.g., “broadcast” type, or cost equaling 10).

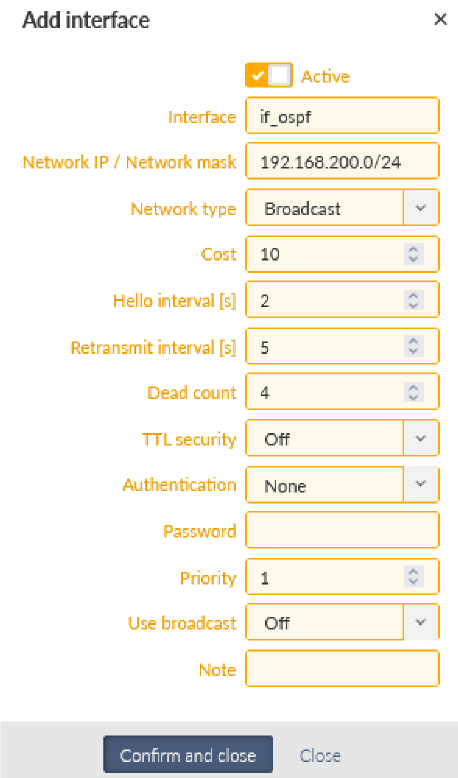


Fig. 7.13: RipEX_C – New OSPF interface under backbone 0.0.0.0 area

After the changes, the setup should look like this:

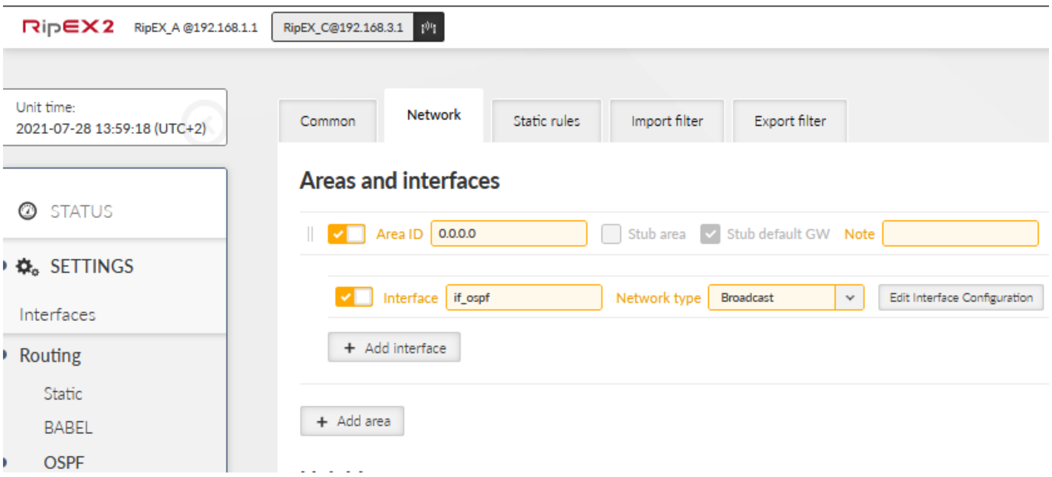


Fig. 7.14: RipEX_C – OSPF network

Configure “Static rules” with the local LAN IP subnet. Set the Metric type to “2” (EXT2) and its metric to 1000. We do not use “OSPF tag” parameter.

EXT2 is used because unit’s LAN is considered to be part of the Babel network.

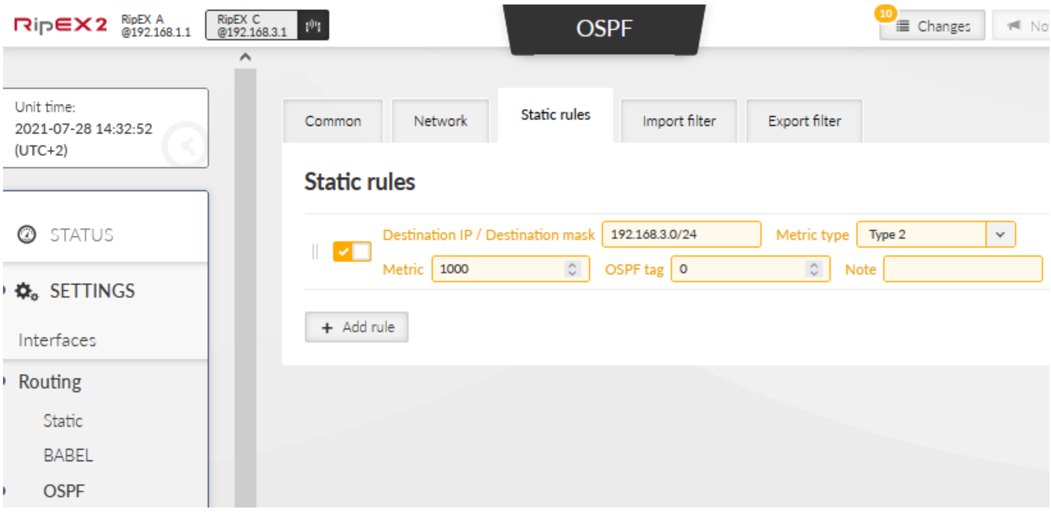


Fig. 7.15: RipEX_C – OSPF Static rules

We need to configure two Import filter rules, the same way as in RipEX_A.

The first one for rules matching EXT2 and setting the Preference to 150. Keep the order of the rules.

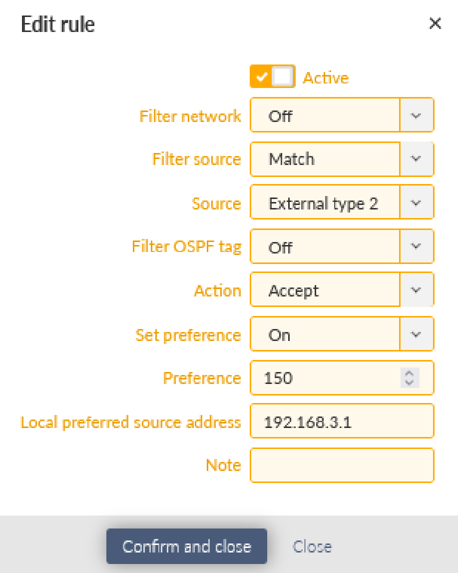


Fig. 7.16: RipEX_C – 1st OSPF Import filter rule

A 2nd rule is for other traffic and setting the Preference to 250 (higher/better than Babel).

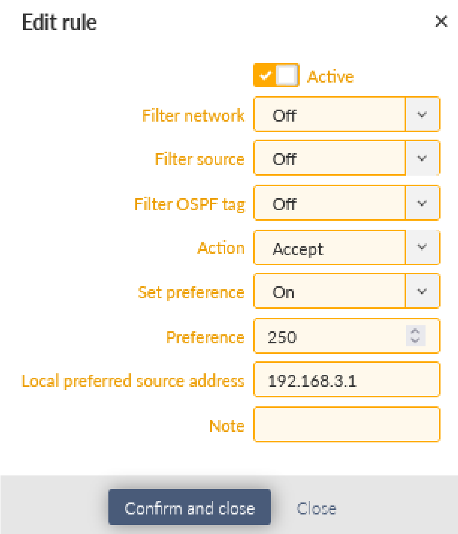


Fig. 7.17: RipEX_C – 2nd OSPF Import filter rule

And the last OSPF setting is Export filter rule. It is the same as in RipEX_A, but we set the Metric to be 2000 so this ASBR/Router is considered as a backup one (RipEX_A is a preferred ASBR).

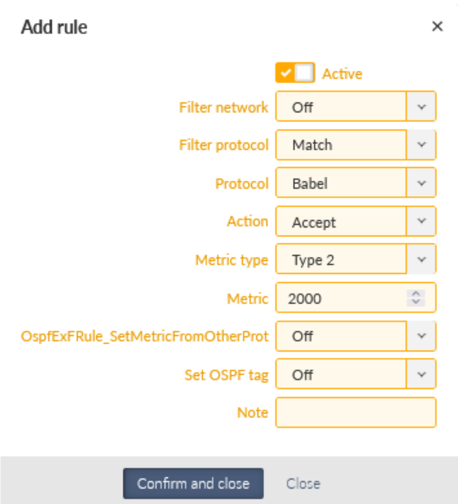


Fig. 7.18: RipEX_C – OSPF external filter rule

Go to the Routing – Babel menu and select an Export filter tab. Add a new rule which exports rules from OSPF with EXT1 type and sets a metric to 2000 (backup router, higher than any sum of metrics in Babel network).

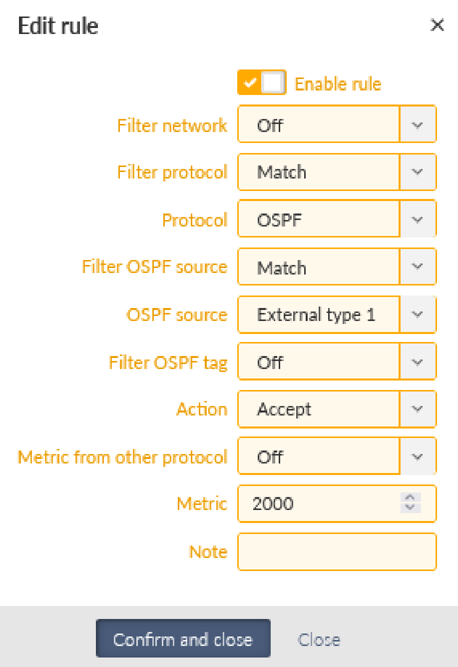


Fig. 7.19: RipEX_C – Babel Export filter rule

Save the changes and configure the last unit – RipEX_D.

7.4. RipEX_D Configuration

Again, we need to start in Interfaces – Ethernet menu. Configure three network interfaces.

ETH1 + ETH2 bridge - 192.168.4.1/24

ETH3 ospf1 - 192.168.100.2/24

ETH4 ospf2 - 192.168.200.2/24

If not yet done, connect ETH3 port to RipEX_A ETH4; and connect ETH4 to RipEX_C ETH4.

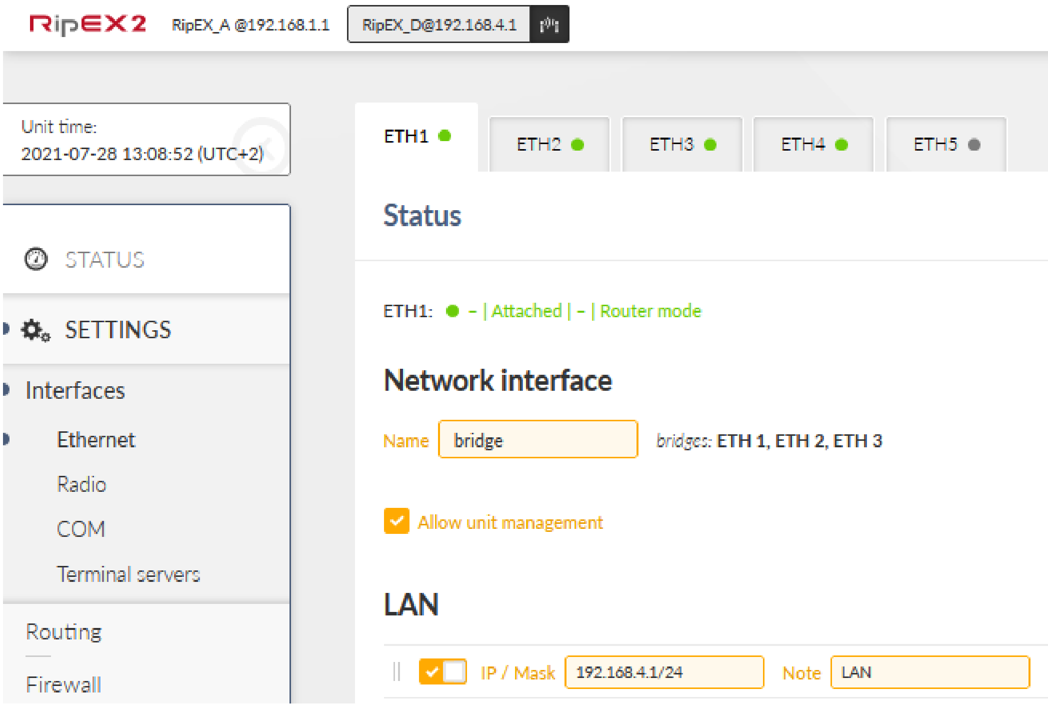


Fig. 7.20: RipEX_D – ETH1 configuration

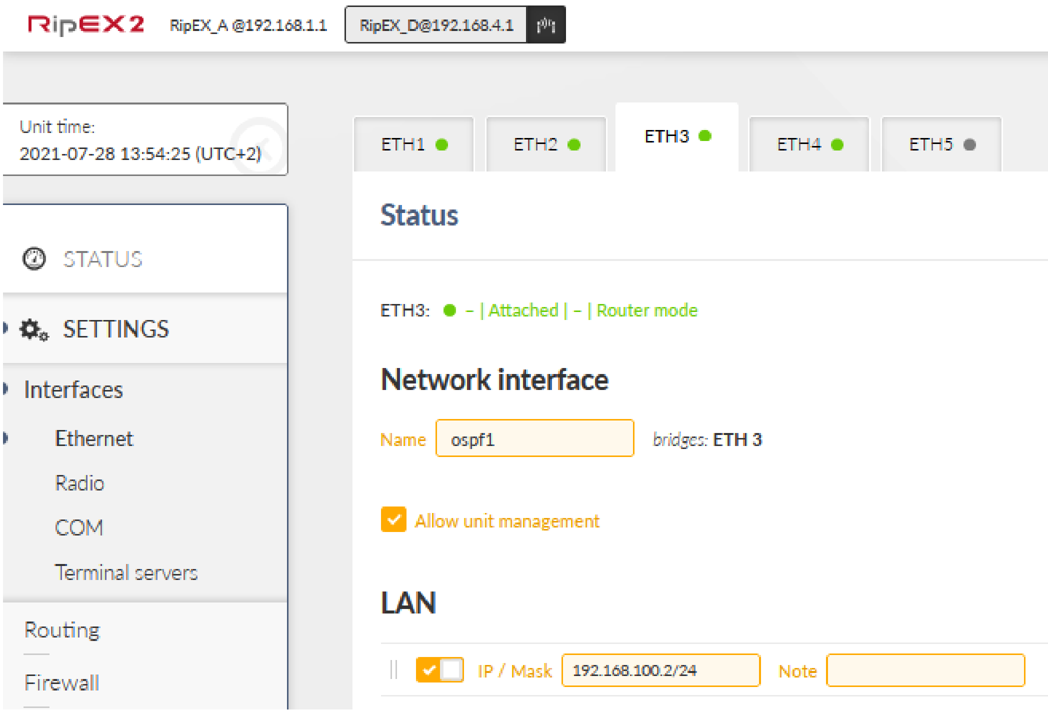


Fig. 7.21: RipEX_D – ETH3 “ospf1” interface

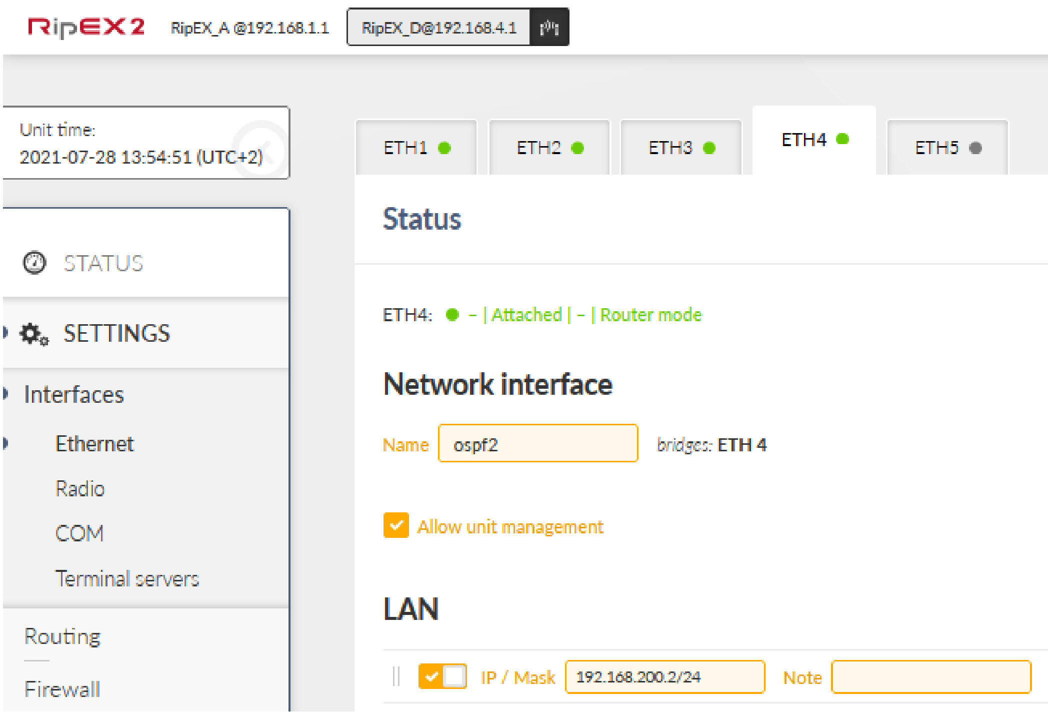


Fig. 7.22: RipEX_D – ETH4 “ospf2” interface

Go to the Routing – OSPF menu and go to the Network tab. Edit the 1st interface and add a 2nd interface. See the non-default values.

- if_ospf1

Type	broadcast
Network IP/mask	192.168.100.0/24
Hello interval [s]	2

- if_ospf2

Type	broadcast
Network IP/mask	192.168.200.0/24
Hello interval [s]	2

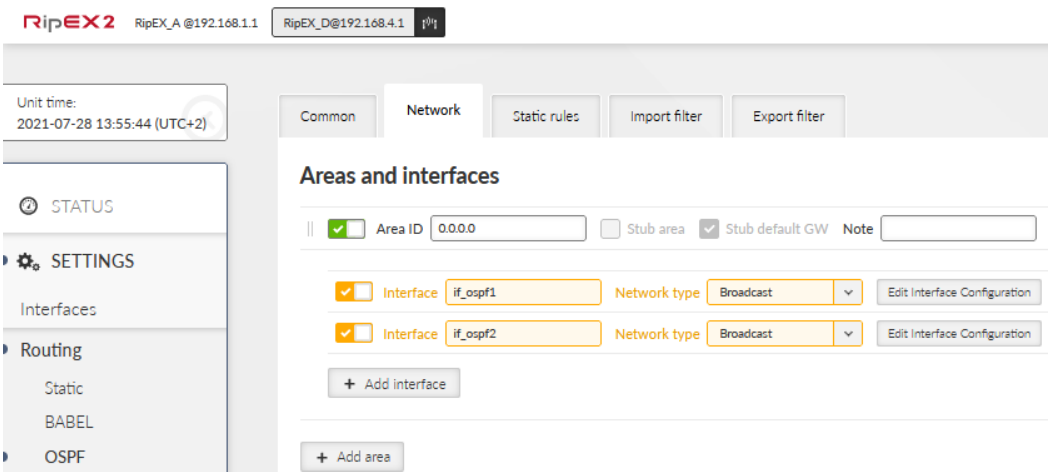


Fig. 7.23: RipEX_D – OSPF Network configuration

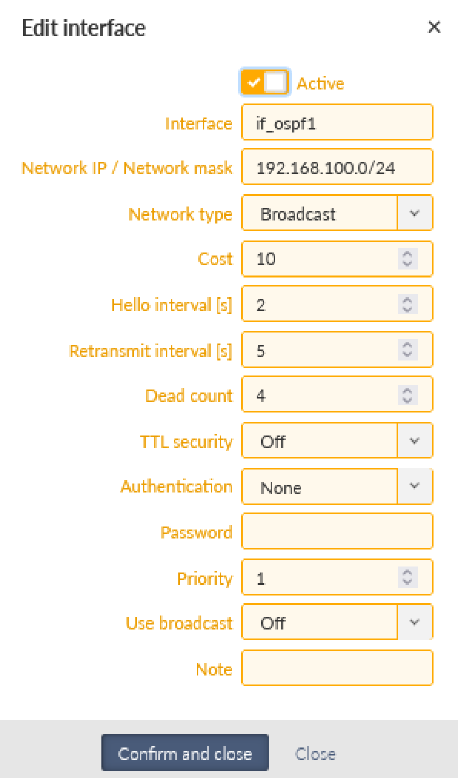


Fig. 7.24: RipEX_D – OSPF interface (if_ospf1)

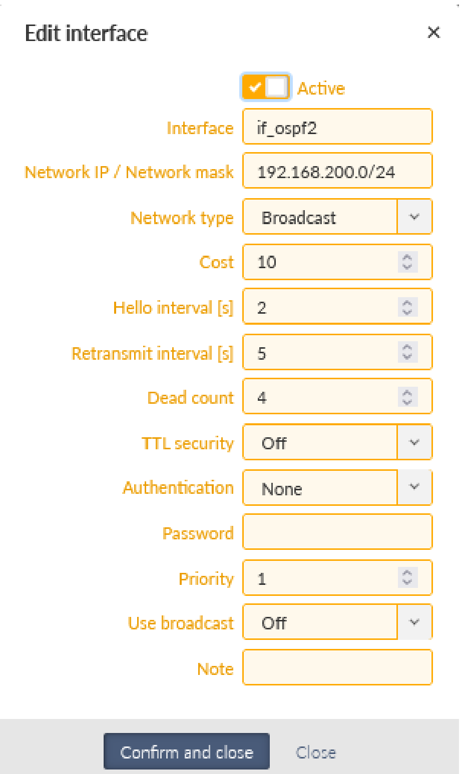


Fig. 7.25: RipEX_D – OSPF interface (if_ospf2)

Change the tab to “Static rules” and make sure there is just one rule for 192.168.4.0/24, Type 1, Metric 1000.

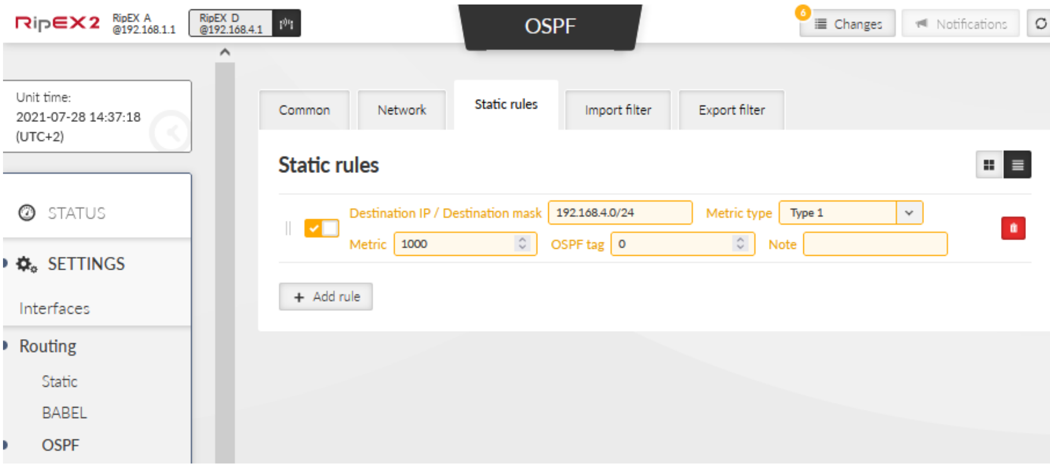


Fig. 7.26: RipEX_D – OSPF Static rules

There are no special changes in other OSPF tabs. Babel is not configured in RipEX_D. Save the changes. We recommend rebooting all RipEX2 units after final configuration to speed things up.

7.5. Diagnostics and Testing

Compared to previous example, communication from RipEX_D can use either a link via RipEX_A (primary ASBR), or via RipEX_C (backup ASBR).

From the other end, RipEX_B can communicate with RipEX_D via RipEX_A (primary), or via RipEX_C (backup).

So, first of all, check the DIAGNOSTICS – Routing menu in all the units. Afterwards, check if you can reach every LAN IP addresses from each unit. Eventually, remove Ethernet cable between RipEX_A and RipEX_D and check if the communication starts via RipEX_C. Connect the cable back and check if/when it goes back to primary link. Ping can be running from RipEX_B to RipEX_D, or vice versa (for example).

You can also run ping between RipEX_A and RipEX_C – it uses the Radio link. Try to remove antennas and check if the ping starts to use an Ethernet link via RipEX_D.

7.5.1. Checking Routing tables

Check RipEX_D if OSPF is established successfully (State: Full) with both other RipEX2 units, Router IDs 1.1.1.1 and 3.3.3.3 with correct IP address. Note external metrics for 192.168.1.0/24 and 192.168.2.0/24, i.e., metric 1000 for routes via RipEX_A and metric 2000 for routes via RipEX_C.

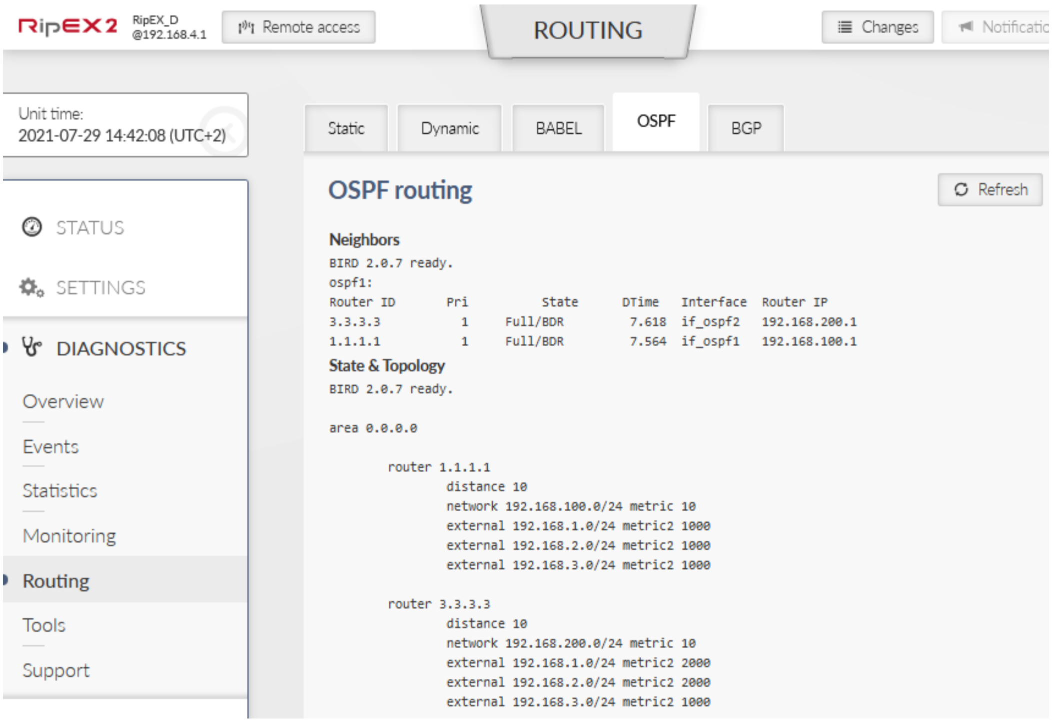


Fig. 7.27: RipEX_D – OSPF Routing diagnostics

Check Babel routes in RipEX_B. Especially check a route 192.168.4.0/24. If all is configured and connected physically correctly, primary link should go via RipEX_A with Metric equal to 1100. A candidate route should be routable via RipEX_C with Metric equal to 2100 (100 = one radio hop, 2000 = exported EXT2 Babel metric from RipEX_C).

If both units have correct routes, it is highly probable that routes in ASBR routers are also correct.

You can e.g., check that both routers have a route to each other primarily via Radio link and a backup link via RipEX_D (over Ethernet).

```
192.168.1.0/24       unicast [babel1 12:40:48.052 from fe80::202:a9ff:fe20:6f9] (210/100) [00:00:00:0
        via 10.10.10.1 on radio
        Type: Babel univ
        Babel.metric: 100
        Babel.router_id: 00:00:00:00:01:01:01:01
        Kernel.prefsrc: 192.168.3.1
                     unicast [ospf1 2021-07-28] E2 (150/20/1000) [1.1.1.1]
        via 192.168.200.2 on if_ospf
        Type: OSPF-E2 univ
        Kernel.prefsrc: 192.168.3.1
        OSPF.metric1: 20
        OSPF.metric2: 1000
        OSPF.tag: 0x00000000
        OSPF.router_id: 1.1.1.1
```

Fig. 7.28: RipEX_C – Dynamic routing diagnostics

Note that Babel Preference to 192.168.1.0/24 via 10.10.10.1 is 210, whereas OSPF Preference is 150. Higher the Preference, more preferred route. The OSPF Metric1 is 20 (2× 10 for OSPF) and Metric2 is 1000 (EXT2 – i.e., it does not sum metrics on the path compared to a sum for EXT1 metrics).

7.5.2. Tools – ICMP ping and Routing

Choose from what device you ping what IP addresses. We check pings from RipEX_D to

- 192.168.1.1
- 192.168.2.1
- 192.168.3.1

Go to the DIAGNOSTICS – Tools – Routing. Check what gateway is used for a particular IP. We should see 192.168.100.1 for 192.168.1.1 and 192.168.2.1. A gateway 192.168.200.2 should be seen for 192.168.3.1.

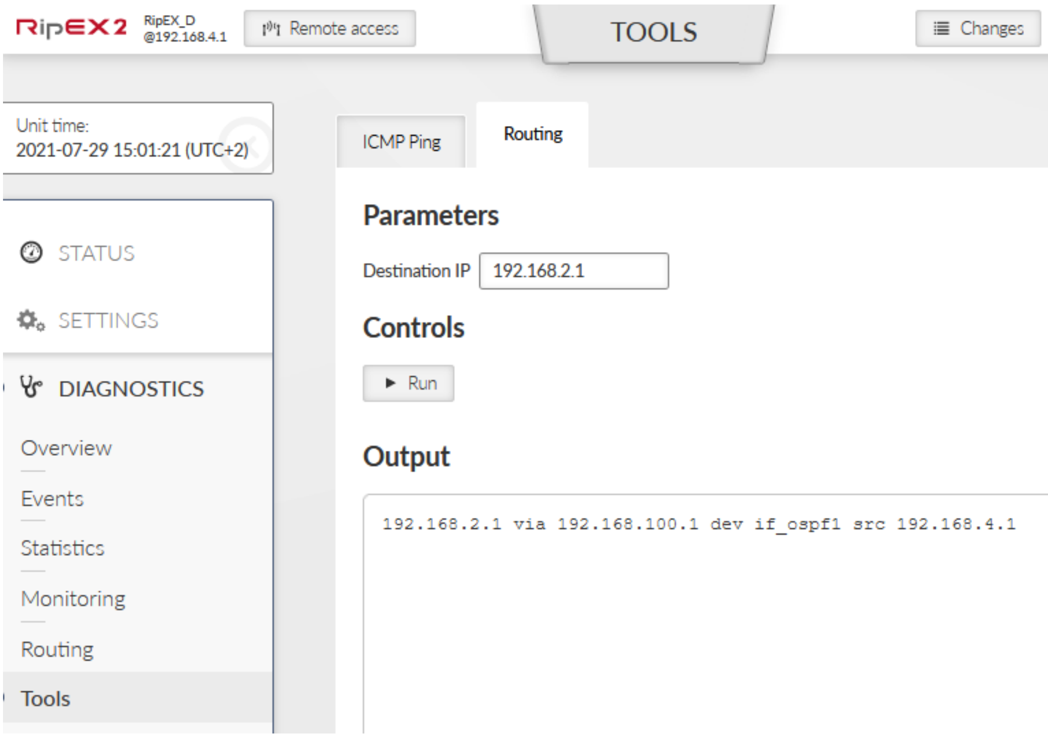


Fig. 7.29: RipEX_D – Routing check to remote networks

Select ICMP ping tab and ping the mentioned IP addresses.

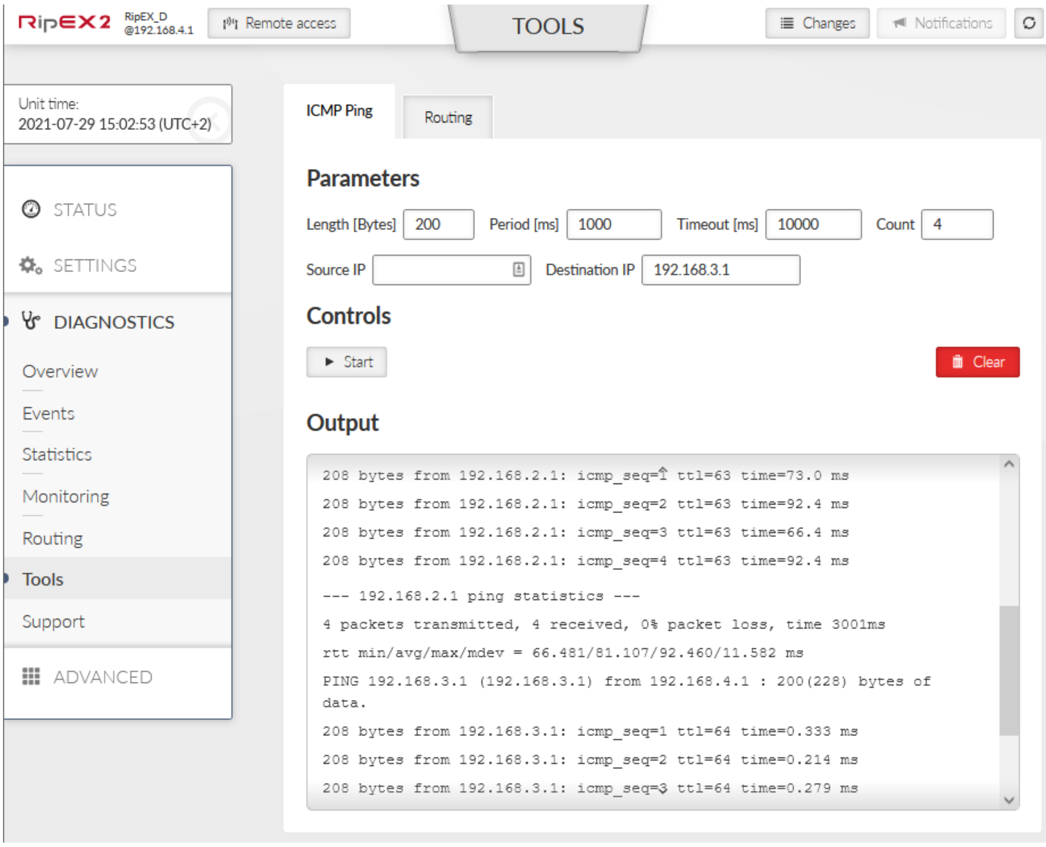


Fig. 7.30: RipEX_D – ICMP ping to remote IP addresses

7.5.3. Testing Ethernet failures

You can also ping from connected PC/laptop to remote IP addresses rather than a ping started in web interface. As the 1st test, we start a ping to 192.168.2.1 from PC 192.168.4.254 connected locally to RipEX_D.

Fig. 7.31: RipEX_D – Highlighted dynamic route to 192.168.2.0/24 via RipEX_A

Once the ping is running successfully, disconnect the Ethernet cable between RipEX_D and RipEX_A. Keep checking current dynamic routing rules continuously and wait for ICMP ping to start working again (via RipEX_C).

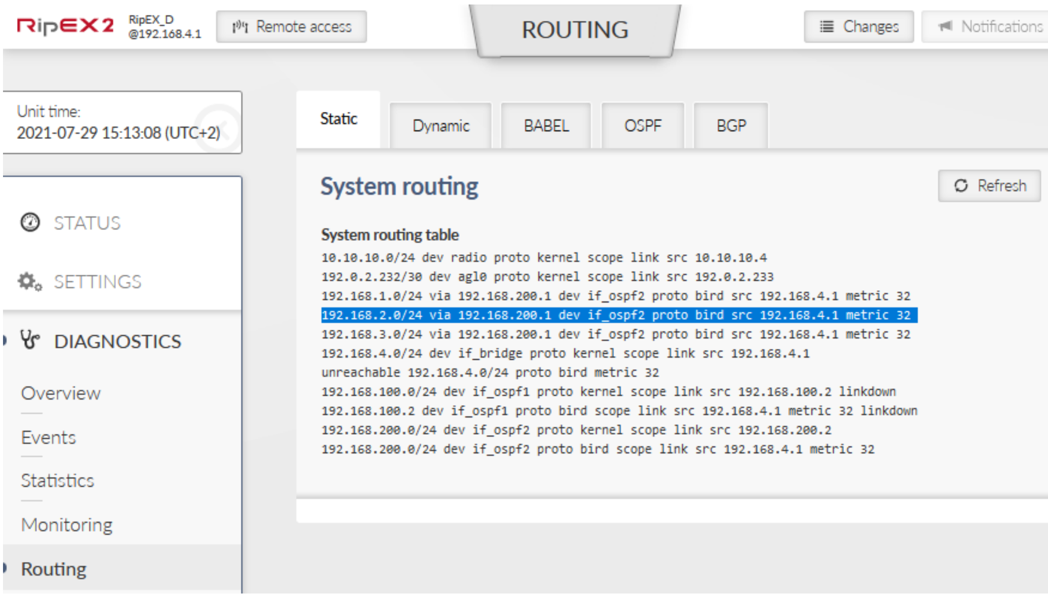


Fig. 7.32: RipEX_D – Highlighted dynamic route to 192.168.2.0/24 via RipEX_C

Because the OSPF timing is set to be very fast, you should also see a rapid (within several seconds) change in the mentioned routing.

Connect the Ethernet cable back and see when the communication comes back via primary ASBR RipEX_A.

7.5.4. Testing Radio failures

Connect locally to RipEX_A, e.g., using 192.168.1.254/24 IP address and static routes to remote networks.

Start a ping to 192.168.3.1 (RipEX_C). The ping should be running over the Radio channel and RTT should be in tens/hundreds of milliseconds (based on selected modulation).

```
C:\Windows\system32>ping 192.168.3.1 -t

Pinging 192.168.3.1 with 32 bytes of data:
Reply from 192.168.3.1: bytes=32 time=40ms TTL=63
Reply from 192.168.3.1: bytes=32 time=47ms TTL=63
Reply from 192.168.3.1: bytes=32 time=40ms TTL=63
Reply from 192.168.3.1: bytes=32 time=47ms TTL=63
Reply from 192.168.3.1: bytes=32 time=47ms TTL=63
```

Fig. 7.33: RipEX_A – Ping from laptop to RipEX_C

Remove RipEX_A and RipEX_C antennas to stop Radio traffic communication and keep checking Dynamic routing in RipEX_A diagnostics.

Neighbors

```
BIRD 2.0.7 ready.
babel1:
IP address                Interface  Metric Routes Hellos Expires
fe80::202:a9ff:fe20:ae3   radio         160      3     10   1.164
fe80::202:a9ff:fe20:789   radio         177      3      9  18.274
```

Routes

```
BIRD 2.0.7 ready.
babel1:
Prefix               Nexthop                      Interface Metric F Seqno Expires
192.168.2.0/24       10.10.10.2                   radio        177 *     3 147.276
192.168.2.0/24       10.10.10.3                   radio        260       3 181.690
192.168.3.0/24       10.10.10.3                   radio        160 *     3 181.690
192.168.3.0/24       10.10.10.2                   radio        277       3 147.276
192.168.4.0/24       10.10.10.3                   radio       2160 *     3 181.690
192.168.4.0/24       10.10.10.2                   radio      65535       3 147.986
```

Fig. 7.34: RipEX_A – Babel metrics go down for the Radio channel

Also keep an eye on running ping. The communication should stop working until a route via Ethernet (RipEX_D) is used an RTT drops to units of milliseconds. This process is much slower due to Babel settings on the Radio channel (30 seconds Hello intervals, …) and big difference in metrics (100 for one Babel hop, whereas 1000 or 2000 metrics for OSPF).

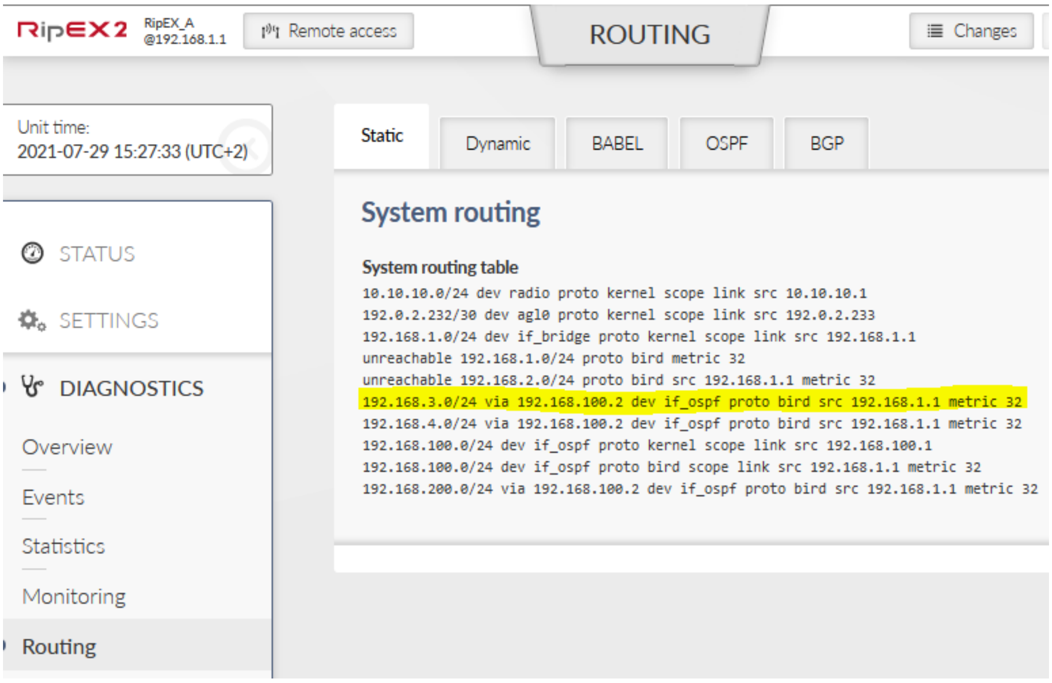


Fig. 7.35: RipEX_A – Dynamic routing using Ethernet (OSPF) connection to RipEX_C

Ping starts working via Ethernet.

```
Reply from 192.168.3.1: bytes=32 time<1ms TTL=62
Reply from 192.168.3.1: bytes=32 time<1ms TTL=62
Reply from 192.168.3.1: bytes=32 time<1ms TTL=62
Reply from 192.168.3.1: bytes=32 time=4ms TTL=62
Reply from 192.168.3.1: bytes=32 time<1ms TTL=62
Reply from 192.168.3.1: bytes=32 time<1ms TTL=62
```

Fig. 7.36: RipEX_A – RTT drops down

OSPF works fine, whereas Babel has no route to the destination.

```
192.168.3.0/24       unicast [ospf1 13:14:21.394] E2 (150/20/1000) [3.3.3.3]
        via 192.168.100.2 on if_ospf
        Type: OSPF-E2 univ
        Kernel.prefsrc: 192.168.1.1
        OSPF.metric1: 20
        OSPF.metric2: 1000
        OSPF.tag: 0x00000000
        OSPF.router_id: 3.3.3.3
                     unreachable [babel1 13:27:06.484] (1/0) [00:00:00:00:00:00:00:00]
        Type: Babel univ
        Kernel.prefsrc: 192.168.1.1
```

Fig. 7.37: RipEX_A – Babel route is unreachable

Connect antennas back and see if and when it comes back. This operation should take less time than going from primary to backup route.

For the whole process, you can also check Monitoring and Statistics pages of RipEX2 Diagnostics.

8. Hints and Tips

8.1. Throughput, speed

As already mentioned within examples, consider Hello intervals together with SCADA protocol requirements. Using 4 RipEX2 units, these were measured values on the RF channel (just several examples):

- Hello interval: 5 seconds, Multiplier: 2, ~ 1 pps, 792 bps
  - Path switching usually within several tens of seconds, up to 2 minutes
- Hello interval: 15 seconds, Multiplier: 4, ~ 0.3 pps, 224 bps
- Hello interval: 20 seconds, Multiplier: 3, ~ 0.2 pps, 168 bps
  - Path switching usually 1-3 minutes
- Hello interval: 30 seconds, Multiplier: 4, ~ 0.1 pps, 102 bps
  - Path switching between approx. 30 seconds and 5 minutes

8.2. Static path setup

It might be beneficial to statically define those packets should primarily go over repeater “A” and only if this link fails, use repeater “B”, or even direct link. Babel (and OSPF/BGP) work in background and logic is based on metrics and particular algorithms. It is not possible to manually define it this way as with RipEX Backup paths. Only the received cost can be set to higher or lower values, but per unit/interface, not per neighboring IP. E.g., one repeater can have Rx cost set to 100, whereas second one 200. This can prioritize the first one.

It is also possible that while topology changes, the metric can change up and down for multiple paths. It can switch to turned off path again, because even other metrics are being increased. Eventually, the path over turned off repeater gets a metric of 65535 and then disappears completely.

8.3. Advanced configurations

For any other advanced setup, contact our technical support *support@racom.eu*[1].

[1] mailto:support@racom.eu

Revision History

Revision 1.0 2021-10-01

First issue

Revision 1.1 2024-08-28

Added *Chapter 2, Mesh topology with Radio and Relay filters*