Gluon Documentation

Release 2016.1.3

Project Gluon

March 31, 2016

Contents

Gluon is a modular framework for creating OpenWrt-based firmwares for wireless mesh nodes. Several Freifunk communities in Germany use Gluon as the foundation of their Freifunk firmwares.

CHAPTER 1

User Documentation

1.1 Getting Started

1.1.1 Selecting the right version

Gluon's releases are managed using Git tags. If you are just getting started with Gluon we recommend to use the latest stable release of Gluon.

Take a look at the list of gluon releases and notice the latest release, e.g. *v2016.1.3*. Always get Gluon using git and don't try to download it as a Zip archive as the archive will be missing version information.

Please keep in mind that there is no "default Gluon" build; a site configuration is required to adjust Gluon to your needs. Due to new features being added (or sometimes being removed) the format of the site configuration changes slightly between releases. Please refer to our release notes for instructions to update an old site configuration to a newer release of Gluon.

An example configuration can be found in the Gluon repository at *docs/site-example/*.

1.1.2 Dependencies

To build Gluon, several packages need to be installed on the system. On a freshly installed Debian Wheezy system the following packages are required:

- *git* (to get Gluon and other dependencies)
- *subversion*
- *python* (Python 3 doesn't work)
- *build-essential*
- *gawk*
- *unzip*
- *libncurses-dev* (actually *libncurses5-dev*)
- *libz-dev* (actually *zlib1g-dev*)
- *libssl-dev*

1.1.3 Building the images

To build Gluon, first check out the repository. Replace *RELEASE* with the version you'd like to checkout, e.g. *v2016.1.3*.

```
git clone https://github.com/freifunk-gluon/gluon.git gluon -b RELEASE
```

This command will create a directory named *gluon/*. It might also tell a scary message about being in a *detached state*. **Don't panic!** Everything's fine. Now, enter the freshly created directory:

```
cd gluon
```

It's time to add (or create) your site configuration. If you already have a site repository, just clone it:

```
git clone https://github.com/freifunk-duckburg/site-ffdb.git site
```

If you want to build a new site, create a new git repository *site/*:

```
mkdir site
cd site
git init
```

Copy *site.conf*, *site.mk* and *i18n* from *docs/site-example*:

```
cp ../docs/site-example/site.conf .
cp ../docs/site-example/site.mk .
cp -r ../docs/site-example/i18n .
```

Edit these files as you see fit and commit them into the site repository. Extensive documentation about the site configuration can be found at: Site configuration. The site directory should always be a git repository by itself; committing site-specific files to the Gluon main repository should be avoided, as it will make updates more complicated.

Next go back to the top-level Gluon directory and build Gluon:

```
cd ..
make update                        # Get other repositories used by Gluon
make GLUON_TARGET=ar71xx-generic   # Build Gluon
```

When calling make, the OpenWrt build environment is prepared/updated. In case of errors read the messages carefully and try to fix the stated issues (e.g. install tools not available yet).

`ar71xx-generic` is the most common target and will generate images for most of the supported hardware. To see a complete list of supported targets, call `make` without setting `GLUON_TARGET`.

You should reserve about 10GB of disk space for each *GLUON_TARGET*.

The built images can be found in the directory *output/images*. Of these, the *factory* images are to be used when flashing from the original firmware a device came with, and *sysupgrade* is to upgrade from other versions of Gluon or any other OpenWrt-based system.

Note: The images for some models are identical; to save disk space, symlinks are generated instead of multiple copies of the same image. If your webserver's configuration prohibits following symlinks, you can use the following command to resolve these links while copying the images:

```
cp -rL output/images /var/www
```

Cleaning the build tree

There are two levels of *make clean*:

```
make clean GLUON_TARGET=ar71xx-generic
```

will ensure all packages are rebuilt for a single target; this is what you normally want to do after an update.

```
make dirclean
```

will clean the entire tree, so the toolchain will be rebuilt as well, which is not necessary in most cases, and will take a while.

1.1.4 opkg repositories

Gluon is mostly compatible with OpenWrt, so the normal OpenWrt package repositories can be used for Gluon as well. It is advisable to setup a mirror or reverse proxy reachable over IPv6 and add it to `site.conf` as http://downloads.openwrt.org/ does not support IPv6.

This is not true for kernel modules; the Gluon kernel is incompatible with the kernel of the default OpenWrt images. Therefore, Gluon will not only generate images, but also an opkg repositoy containing all kernel modules provided by OpenWrt/Gluon for the kernel of the generated images.

Signing keys

Gluon does not support HTTPS for downloading packages; fortunately, opkg deploys public-key cryptography to ensure package integrity.

The Gluon images will contain two public keys: the official OpenWrt signing key (to allow installing userspace packages) and a Gluon-specific key (which is used to sign the generated module repository).

By default, Gluon will handle the generation and handling of the keys itself. When making firmware releases based on Gluon, it might make sense to store the keypair, so updating the module repository later is possible.

The location the keys are stored at and read from can be changed (see *Environment variables*). To only generate the keypair at the configured location without doing a full build, use `make create-key`.

1.1.5 Environment variables

Gluon's build process can be controlled by various environment variables.

GLUON_SITEDIR Path to the site configuration. Defaults to `site`.

GLUON_BUILDDIR Working directory during build. Defaults to `build`.

GLUON_OPKG_KEY Path key file used to sign the module opkg repository. Defaults to `$(GLUON_BULDDIR)/gluon-opkg-key`.

The private key will be stored as `$(GLUON_OPKG_KEY)`, the public key as `$(GLUON_OPKG_KEY).pub`.

GLUON_OUTPUTDIR Path where output files will be stored. Defaults to `output`.

GLUON_IMAGEDIR Path where images will be stored. Defaults to `$(GLUON_OUTPUTDIR)/images`.

GLUON_MODULEDIR Path where the kernel module opkg repository will be stored. Defaults to `$(GLUON_OUTPUTDIR)/modules`.

So all in all, to update and rebuild a Gluon build tree, the following commands should be used (repeat the `make clean` and `make` for all targets you want to build):

```
git pull
(cd site && git pull)
make update
make clean GLUON_TARGET=ar71xx-generic
make GLUON_TARGET=ar71xx-generic
```

1.2 Site configuration

The `site` consists of the files `site.conf` and `site.mk`. In the first community based values are defined, which both are processed during the build process and runtime. The last is directly included in the make process of Gluon.

1.2.1 Configuration

The `site.conf` is a lua dictionary with the following defined keys.

hostname_prefix A string which shall prefix the default hostname of a device.

site_name The name of your community.

site_code The code of your community. It is good practice to use the TLD of your community here.

prefix4 The IPv4 Subnet of your community mesh network in CIDR notation, e.g.

```
    prefix4 = '10.111.111.0/18'
```

prefix6 The IPv6 subnet of your community mesh network, e.g.

```
    prefix6 = 'fdca::ffee:babe:1::/64'
```

timezone The timezone of your community live in, e.g.

```
    -- Europe/Berlin
    timezone = 'CET-1CEST,M3.5.0,M10.5.0/3'
```

ntp_server List of NTP servers available in your community or used by your community, e.g.:

```
    ntp_servers = {'1.ntp.services.ffeh','2.tnp.services.ffeh'}
```

opkg [optional] `opkg` package manager configuration.

There are two optional fields in the `opkg` section:

- `openwrt` overrides the default OpenWrt repository URL
- `extra` specifies a table of additional repositories (with arbitrary keys)

```
    opkg = {
      openwrt = 'http://opkg.services.ffeh/openwrt/%n/%v/%S/packages',
      extra = {
        modules = 'http://opkg.services.ffeh/modules/gluon-%GS-%GR/%S',
      },
    }
```

There are various patterns which can be used in the URLs:

- `%n` is replaced by the OpenWrt version codename (e.g. "chaos_calmer")
- `%v` is replaced by the OpenWrt version number (e.g. "15.05")
- `%S` is replaced by the target architecture (e.g. "ar71xx/generic")

- %GS is replaced by the Gluon site code (as specified in site.conf)
- %GV is replaced by the Gluon version
- %GR is replaced by the Gluon release (as specified in site.mk)

regdom [optional] The wireless regulatory domain responsible for your area, e.g.:

```
regdom = 'DE'
```

Setting regdom in mandatory if wifi24 or wifi5 is defined.

wifi24 [optional] WLAN configuration for 2.4 GHz devices. channel must be set to a valid wireless channel for your radio.

There are currently three interface types available. You many choose to configure any subset of them:

- ap creates a master interface where clients may connect
- mesh creates an 802.11s mesh interface with forwarding disabled
- ibss creates an ad-hoc interface

Each interface may be disabled by setting disabled to true. This will only affect new installations. Upgrades will not changed the disabled state.

ap requires a single parameter, a string, named ssid which sets the interface's ESSID.

mesh requires a single parameter, a string, named id which sets the mesh id.

ibss requires two parametersr: ssid (a string) and bssid (a MAC). An optional parameter vlan (integer) is supported.

Both mesh and ibss accept an optional mcast_rate (kbit/s) parameter for setting the default multicast datarate.

```
wifi24 = {
  channel = 11,
  ap = {
    ssid = 'entenhausen.freifunk.net',
  },
  mesh = {
    id = 'entenhausen-mesh',
    mcast_rate = 12000,
  },
  ibss = {
    ssid = 'ff:ff:ff:ee:ba:be',
    bssid = 'ff:ff:ff:ee:ba:be',
    mcast_rate = 12000,
  },
},
```

wifi5 [optional] Same as *wifi24* but for the 5Ghz radio.

next_node [package] Configuration of the local node feature of Gluon

```
next_node = {
  ip4 = '10.23.42.1',
  ip6 = 'fdca:ffee:babe:1::1',
  mac = 'ca:ff:ee:ba:be:00'
}
```

mesh [optional] Options specific to routing protocols.

At the moment, only the batman_adv routing protocol has such options:

The optional value `gw_sel_class` sets the gateway selection class. The default class 20 is based on the link quality (TQ) only, class 1 is calculated from both the TQ and the announced bandwidth.

```
mesh = {
  batman_adv = {
    gw_sel_class = 1,
  },
}
```

fastd_mesh_vpn Remote server setup for the fastd-based mesh VPN.

The *enabled* option can be set to true to enable the VPN by default.

If *configurable* is set to *false* or unset, the method list will be replaced on updates with the list from the site configuration. Setting *configurable* to *true* will allow the user to add the method `null` to the beginning of the method list or remove `null` from it, and make this change survive updates. Setting *configurable* is necessary for the package *gluon-luci-mesh-vpn-fastd*, which adds a UI for this configuration.

In any case, the `null` method should always be the first method in the list if it is supported at all. You should only set *configurable* to *true* if the configured peers support both the `null` method and methods with encryption.

```
fastd_mesh_vpn = {
  methods = {'salsa2012+umac'},
  -- enabled = true,
  -- configurable = true,
  mtu = 1280,
  groups = {
    backbone = {
      -- Limit number of connected peers from this group
      limit = 1,
      peers = {
        peer1 = {
          key = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX',
          -- Having multiple domains prevents SPOF in freifunk.net
          remotes = {
            'ipv4 "vpn1.entenhausen.freifunk.net" port 10000',
            'ipv4 "vpn1.entenhausener-freifunk.de" port 10000',
          },
        },
        peer2 = {
          key = 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX',
          -- You can also omit the ipv4 to allow both connection via ipv4 and ipv6
          remotes = {'"vpn2.entenhausen.freifunk.net" port 10000'},
        },
      },
      -- Optional: nested peer groups
      -- groups = {
      --   lowend_backbone = {
      --     limit = 1,
      --     peers = ...
      --   },
      -- },
    },
    -- Optional: additional peer groups, possibly with other limits
    -- peertopeer = {
    --   limit = 10,
    --   peers = { ... },
    -- },
  },
  bandwidth_limit = {
    -- The bandwidth limit can be enabled by default here.
    enabled = false,

    -- Default upload limit (kbit/s).
    egress = 200,

    -- Default download limit (kbit/s).
    ingress = 3000,
  },
}
```

mesh_on_wan [optional] Enables the mesh on the WAN port (`true` or `false`).

mesh_on_lan [optional] Enables the mesh on the LAN port (`true` or `false`).

autoupdater [package] Configuration for the autoupdater feature of Gluon.

```
  autoupdater = {
    branch = 'stable',
    branches = {
      stable = {
        name = 'stable',
        mirrors = {
          'http://[fdca:ffee:babe:1::fec1]/firmware/stable/sysupgrade/',
          'http://autoupdate.entenhausen.freifunk.net/firmware/stable/sysupgrade/',
        },
        -- Number of good signatures required
        good_signatures = 2,
        pubkeys = {
          'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX', -- someguy
          'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX', -- someother
        }
      }
    }
  }
```

roles [optional] Optional role definitions. Nodes will announce their role inside the mesh. This will allow in the backend to distinguish between normal, backbone and service nodes or even gateways (if they advertise that role). It is up to the community which roles to define. See the section below as an example. `default` takes the default role which is set initially. This value should be part of `list`. If you want node owners to change the role via config mode add the package `gluon-luci-node-role` to `site.mk`.

The strings to display in the LuCI interface can be configured per language in the `i18n/en.po`, `i18n/de.po`, etc. files of the site repository using message IDs like `gluon-luci-node-role:role:node` and `gluon-luci-node-role:role:backbone`.

```
  roles = {
    default = 'node',
    list = {
      'node',
      'test',
      'backbone',
      'service',
    },
  },
```

setup_mode [package] Allows skipping setup mode (config mode) at first boot when attribute `skip` is set to `true`. This is optional and may be left out.

```
    setup_mode = {
      skip = true,
    },
```

legacy [package] Configuration for the legacy upgrade path. This is only required in communities upgrading from Lübeck's LFF-0.3.x.

```
    legacy = {
           version_files = {'/etc/.freifunk_version_keep', '/etc/.eff_version_keep'},
           old_files = {'/etc/config/config_mode', '/etc/config/ffeh', '/etc/config/freifunk'},
           config_mode_configs = {'config_mode', 'ffeh', 'freifunk'},
           fastd_configs = {'ffeh_mesh_vpn', 'mesh_vpn'},
           mesh_ifname = 'freifunk',
           tc_configs = {'ffki', 'freifunk'},
           wifi_names = {'wifi_freifunk', 'wifi_freifunk5', 'wifi_mesh', 'wifi_mesh5'},
    }
```

1.2.2 Packages

The `site.mk` is a Makefile which should define constants involved in the build process of Gluon.

GLUON_SITE_PACKAGES Defines a list of packages which should be installed additionally to the `gluon-core` package.

GLUON_RELEASE The current release version Gluon should use.

GLUON_PRIORITY The default priority for the generated manifests (see the autoupdater documentation for more information).

GLUON_LANGS List of languages (as two-letter-codes) to be included in the web interface. Should always contain `en`.

1.2.3 Config mode texts

The community-defined texts in the config mode are configured in PO files in the `i18n` subdirectory of the site configuration. The message IDs currently defined are:

gluon-config-mode:welcome Welcome text on the top of the config wizard page.

gluon-config-mode:pubkey Information about the public VPN key on the reboot page.

gluon-config-mode:reboot General information shown on the reboot page.

There is a POT file in the site example directory which can be used to create templates for the language files. The command `msginit -l en -i ../../docs/site-example/i18n/gluon-site.pot` can be used from the `i18n` directory to create an initial PO file called `en.po` if the `gettext` utilities are installed.

Note: An empty `msgstr`, as is the default after running `msginit`, leads to the `msgid` being printed as-is. It does *not* hide the whole text, as might be expected.

Depending on the context, you might be able to use comments like `<!-- empty -->` as translations to effectively hide the text.

1.2.4 Examples

site.mk

```
##        gluon site.mk makefile example

##        GLUON_SITE_PACKAGES
#                specify gluon/openwrt packages to include here
#                The gluon-mesh-batman-adv-* package must come first because of the dependency resolu

GLUON_SITE_PACKAGES := \
        gluon-mesh-batman-adv-15 \
        gluon-alfred \
        gluon-respondd \
        gluon-autoupdater \
        gluon-config-mode-autoupdater \
        gluon-config-mode-contact-info \
        gluon-config-mode-core \
        gluon-config-mode-geo-location \
        gluon-config-mode-hostname \
        gluon-config-mode-mesh-vpn \
        gluon-ebtables-filter-multicast \
        gluon-ebtables-filter-ra-dhcp \
        gluon-luci-admin \
        gluon-luci-autoupdater \
        gluon-luci-portconfig \
        gluon-luci-wifi-config \
        gluon-next-node \
        gluon-mesh-vpn-fastd \
        gluon-radvd \
        gluon-setup-mode \
        gluon-status-page \
        haveged \
        iptables \
        iwinfo

##        DEFAULT_GLUON_RELEASE
#                version string to use for images
#                gluon relies on
#                        opkg compare-versions "$1" '>>' "$2"
#                to decide if a version is newer or not.

DEFAULT_GLUON_RELEASE := 0.6+exp$(shell date '+%Y%m%d')


##        GLUON_RELEASE
#                call make with custom GLUON_RELEASE flag, to use your own release version scheme.
#                e.g.:
#                        $ make images GLUON_RELEASE=23.42+5
#                would generate images named like this:
#                        gluon-ff%site_code%-23.42+5-%router_model%.bin

# Allow overriding the release number from the command line
GLUON_RELEASE ?= $(DEFAULT_GLUON_RELEASE)

# Default priority for updates.
GLUON_PRIORITY ?= 0
```

```
# Languages to include
GLUON_LANGS ?= en de
```

site.conf

```
-- This is an example site configuration for Gluon v2016.1.3
--
-- Take a look at the documentation located at
-- http://gluon.readthedocs.org/ for details.
--
-- This configuration will not work as it. You're required to make
-- community specific changes to it!
{
  -- Used for generated hostnames, e.g. freifunk-abcdef123456. (optional)
  -- hostname_prefix = 'freifunk-',

  -- Name of the community.
  site_name = 'Freifunk Entenhausen',

  -- Shorthand of the community.
  site_code = 'ffxx',

  -- Prefixes used within the mesh. Both are required.
  prefix4 = '10.xxx.0.0/20',
  prefix6 = 'fdxx:xxxx:xxxx::/64',

  -- Timezone of your community.
  -- See http://wiki.openwrt.org/doc/uci/system#time_zones
  timezone = 'CET-1CEST,M3.5.0,M10.5.0/3',

  -- List of NTP servers in your community.
  -- Must be reachable using IPv6!
  ntp_servers = {'1.ntp.services.ffxx'},

  -- Wireless regulatory domain of your community.
  regdom = 'DE',

  -- Wireless configuration for 2.4 GHz interfaces.
  wifi24 = {
    -- Wireless channel.
    channel = 1,

    -- ESSID used for client network.
    ap = {
      ssid = 'entenhausen.freifunk.net',
      -- disabled = true, (optional)
    },

    mesh = {
      -- Adjust these values!
      id = 'ffxx-mesh',
      mcast_rate = 12000,
      -- disabled = true, (optional)
    },
  },

  -- Wireless configuration for 5 GHz interfaces.
  -- This should be equal to the 2.4 GHz variant, except
  -- for channel.
  wifi5 = {
    channel = 44,
    ap = {
      ssid = 'entenhausen.freifunk.net',
    },
    mesh = {
      id = 'ffxx-mesh',
      mcast_rate = 12000,
    },
  },

  -- The next node feature allows clients to always reach the node it is
  -- connected to using a known IP address.
  next_node = {
    -- anycast IPs of all nodes
    ip4 = '10.xxx.0.xxx',
    ip6 = 'fdxx:xxxx:xxxx::xxxx',

    -- anycast MAC of all nodes
    mac = 'xe:xx:xx:xx:xx:xx',
  },

  -- Options specific to routing protocols (optional)
  -- mesh = {
    -- Options specific to the batman-adv routing protocol (optional)
    -- batman_adv = {
      -- Gateway selection class (optional)
      -- The default class 20 is based on the link quality (TQ) only,
      -- class 1 is calculated from both the TQ and the announced bandwidth
      -- gw_sel_class = 1,
    -- },
  -- },

  -- Refer to http://fastd.readthedocs.org/en/latest/ to better understand
  -- what these options do.
  fastd_mesh_vpn = {
    -- List of crypto-methods to use.
    methods = {'salsa2012+umac'},
    -- enabled = true,
    -- configurable = true,

    mtu = 1280,
    groups = {
      backbone = {
        -- Limit number of connected peers to reduce bandwidth.
        limit = 1,

        -- List of peers.
        peers = {
          peer1 = {
            key = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',

            -- This is a list, so you might add multiple entries.
            remotes = {'ipv4 "xxx.somehost.invalid" port xxxxxx'},
          },
          peer2 = {
            key = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx',
            -- You can also omit the ipv4 to allow both connection via ipv4 and ipv6
            remotes = {'"xxx.somehost2.invalid" port xxxxx'},
          },
        },

        -- Optional: nested peer groups
        -- groups = {
        --   backbone_sub = {
        --     ...
        --   },
        --   ...
        -- },
      },
      -- Optional: additional peer groups, possibly with other limits
      -- backbone2 = {
      --   ...
      -- },
    },

    bandwidth_limit = {
      -- The bandwidth limit can be enabled by default here.
      enabled = false,

      -- Default upload limit (kbit/s).
      egress = 200,

      -- Default download limit (kbit/s).
      ingress = 3000,
    },
  },

  autoupdater = {
    -- Default branch. Don't forget to set GLUON_BRANCH when building!
    branch = 'stable',

    -- List of branches. You may define multiple branches.
    branches = {
      stable = {
        name = 'stable',

        -- List of mirrors to fetch images from. IPv6 required!
        mirrors = {'http://1.updates.services.ffhl/stable/sysupgrade'},

        -- Number of good signatures required.
        -- Have multiple maintainers sign your build and only
        -- accept it when a sufficient number of them have
        -- signed it.
        good_signatures = 2,

        -- List of public keys of maintainers.
        pubkeys = {
                'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx', -- Alice
                'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx', -- Bob
                'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx', -- Mary
        },
      },
    },
  },

  -- Node roles
  -- roles = {
  --   default = 'node',
  --   list = {
  --     'node',
  --     'test',
  --     'backbone',
  --     'service',
  --   },
  -- },

  -- Skip setup mode (config mode) on first boot
  -- setup_mode = {
  --  skip = true,
  -- },
}
```

i18n/en.po

```
msgid ""
msgstr ""
"Content-Type: text/plain; charset=UTF-8\n"
"Project-Id-Version: PACKAGE VERSION\n"
"PO-Revision-Date: 2016-02-04 14:28+0100\n"
"Last-Translator: David Lutz <kpanic@hirnduenger.de>\n"
"Language-Team: English\n"
"Language: en\n"
"MIME-Version: 1.0\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=2; plural=(n != 1);\n"

msgid "gluon-config-mode:welcome"
msgstr ""
"Welcome to the setup wizard of your new Freifunk Duckburg node. "
"Please fill out the following form and submit it."

msgid "gluon-config-mode:pubkey"
msgstr ""
"<p>This is your Freifunk node's public key. The node won't be able to "
"connect to the mesh VPN until the key has been registered on the Freifunk "
"Duckburg servers. "
"To register, send the key together with your node's name (<em><%=hostname%></em>) to "
"<a href=\"mailto:keys@entenhausen.freifunk.net\">keys@entenhausen.freifunk.net</a>."
"</p>"
"<div class=\"the-key\">"
" # <%= hostname %>"
" <br/>"
"<%= pubkey %>"
"</div>"


msgid "gluon-config-mode:reboot"
msgstr ""
"<p>The node is currently rebooting and will try to connect to other "
"nearby Freifunk nodes after that. "
```

```
"For more information on the Freifunk Duckburg community, have a look at "
"<a href=\"https://entenhausen.freifunk.net/\">our homepage</a>.</p>"
"<p>To get back to this configuration interface, press the reset button for "
"3 seconds during normal operation. The device will then reboot into config "
"mode.</p>"
"<p>Have fun with your node and exploring of the Freifunk network!</p>"
```

i18n/de.po

```
msgid ""
msgstr ""
"Content-Type: text/plain; charset=UTF-8\n"
"Project-Id-Version: PACKAGE VERSION\n"
"PO-Revision-Date: 2015-03-19 20:28+0100\n"
"Last-Translator: Matthias Schiffer <mschiffer@universe-factory.net>\n"
"Language-Team: German\n"
"Language: de\n"
"MIME-Version: 1.0\n"
"Content-Transfer-Encoding: 8bit\n"
"Plural-Forms: nplurals=2; plural=(n != 1);\n"

msgid "gluon-config-mode:welcome"
msgstr ""
"Willkommen zum Einrichtungsassistenten für deinen neuen Entenhausener "
"Freifunk-Knoten. Fülle das folgende Formular deinen Vorstellungen "
"entsprechend aus und sende es ab."

msgid "gluon-config-mode:pubkey"
msgstr ""
"<p>Dies ist der öffentliche Schlüssel deines Freifunk-Knotens. Erst nachdem "
"er auf den Servern des Entenhausener Freifunk-Projektes eingetragen wurde, "
"kann sich dein Knoten mit dem Entenhausener Mesh-VPN verbinden. Bitte "
"schicke dazu diesen Schlüssel und den Namen deines Knotens "
"(<em><%=hostname%></em>) an "
"<a href=\"mailto:keys@entenhausen.freifunk.net\">keys@entenhausen.freifunk.net</a>."
"</p>"
"<div class=\"the-key\">"
" # <%= hostname %>"
" <br/>"
"<%= pubkey %>"
"</div>"

msgid "gluon-config-mode:reboot"
msgstr ""
"<p>Dein Knoten startet gerade neu und wird anschließend versuchen, "
"sich mit anderen Freifunkknoten in seiner Nähe zu "
"verbinden. Weitere Informationen zur "
"Entenhausener Freifunk-Community findest du auf "
"<a href=\"https://entenhausen.freifunk.net/\">unserer Webseite</a>.</p>"
"<p>Um zu dieser Konfigurationsseite zurückzugelangen, drücke im normalen "
"Betrieb für drei Sekunden den Reset-Button. Das Gerät wird dann im Config "
"Mode neustarten.</p>"
"<p>Viel Spaß mit deinem Knoten und der Erkundung von Freifunk!</p>"
```

modules

```
# This file allows specifying additional repositories to use
# when building gluon.
#
# In most cases, it is not required so don't add it.

##        GLUON_SITE_FEEDS
#                for each feed name given, add the corresponding PACKAGES_* lines
#                documented below
#GLUON_SITE_FEEDS='my_own_packages'

##        PACKAGES_$feedname_REPO
#                the  git repository from where to clone the package feed
#PACKAGES_MY_OWN_PACKAGES_REPO=https://github.com/.../my-own-packages.git


##        PACKAGES_$feedname_COMMIT
#                the version/commit of the git repository to clone
#PACKAGES_MY_OWN_PACKAGES_COMMIT=123456789aabcda1a69b04278e4d38f2a3f57e49

##  PACKAGES_$feedname_BRANCH
#   the branch to check out
#PACKAGES_MY_OWN_PACKAGES_BRANCH=my_branch
```

site-repos in the wild

This is a non-exhaustive list of site-repos from various communities:

- site-ffa (Altdorf, Landshut & Umgebung)
- site-ffbs (Braunschweig)
- site-ffhb (Bremen)
- site-ffda (Darmstadt)
- site-ffgoe (Göttingen)
- site-ffhh (Hamburg)
- site-ffhgw (Greifswald)
- site-ffhl (Lübeck)
- site-ffmd (Magdeburg)
- site-ffmwu (Mainz, Wiesbaden & Umgebung)
- site-ffmyk (Mayen-Koblenz)
- site-ffm (München)
- site-ffms (Münsterland)
- site-ffnw (Nordwest)
- site-ffpb (Paderborn)
- site-ffka (Karlsruhe)
- site-ffrl (Rheinland)
- site-ffrg (Ruhrgebiet)

- site-ffs (Stuttgart)
- site-fftr (Trier)

1.3 x86 support

Gluon can run on normal x86 systems, for example virtual machines and VPN boxes. By default, there is no WLAN support on x86 though.

1.3.1 Targets

The following targets for x86 images exist:

x86-generic Generic x86 support with many different ethernet drivers; should run on most x86 systems.

There are three images:

- *generic* (compressed "raw" image, can written to a disk directly or booted with qemu)
- *virtualbox* (VDI image)
- *vmware* (VMDK image)

These images only differ in the image file format, the content is the same. Therefore there is only a single *x86-generic* sysupgrade image instead of three.

Please note that the *x86-generic* image doesn't include VirtIO support, so another virtual NIC like *pcnet32* must be chosen when using VirtualBox.

x86-kvm The *x86-kvm* image uses VirtIO as its harddisk and network driver.

x86-xen_domu The *x86-xen_domu* target contains the necessary drivers for use in Xen.

x86-64 64bit version of *x86-generic*. Also has VirtIO support, so there's no need for an *x86-64-kvm* target.

1.4 Frequently Asked Questions

CHAPTER 2

Features

2.1 Config Mode

When in Config Mode a node will neither participate in the mesh nor connect to the VPN using the WAN port. Instead, it'll offer a web interface on the LAN port to aid configuration of the node.

Whether a node is in Config Mode can be determined by a characteristic blinking sequence of the SYS LED:

2.1.1 Activating Config Mode

Config Mode is automatically entered at the first boot. You can re-enter Config Mode by pressing and holding the RESET/WPS button for about three seconds. The device should reboot (all LEDs will turn of briefly) and Config Mode will be available.

2.1.2 Port Configuration

In general, Config Mode will be offered on the LAN ports. However, there are two practical exceptions:

- Devices with just one network port will run Config Mode on that port.
- Devices with PoE on the WAN port will run Config Mode on the WAN port instead.

2.1.3 Accessing Config Mode

Config Mode can be accessed at http://192.168.1.1. The node will offer DHCP to clients. Should this fail, you may assign an IP from 192.168.1.0/24 to your computer manually.

2.2 Autoupdater

Gluon contains an automatic update system which can be configured in the site configuration.

2.2.1 Building Images

By default, the autoupdater is disabled (as it is usually not helpful to have unexpected updates during development), but it can be enabled by setting the variable GLUON_BRANCH when building to override the default branch set in the set in the site configuration.

A manifest file for the updater can be generated with *make manifest*. A signing script (using ecdsautils) can by found in the *contrib* directory. When creating the manifest, `GLUON_PRIORITY` can be set on the command line, or it can be taken from the `site.mk`.

The priority defines the maximum number of days that may pass between releasing an update and installation of the images. The update probability will start at 0 after the release time mentioned in the manifest and then slowly rise to 1 up to the point when the number of days given by the priority has passed.

The priority may be an integer or a decimal fraction.

A fully automated nightly build could use the following commands:

```
git pull
(cd site && git pull)
make update
make clean
make -j5 GLUON_TARGET=ar71xx-generic GLUON_BRANCH=experimental
make manifest GLUON_BRANCH=experimental
contrib/sign.sh $SECRETKEY output/images/sysupgrade/experimental.manifest

rm -rf /where/to/put/this/experimental
cp -r output/images /where/to/put/this/experimental
```

2.2.2 Infrastructure

We suggest to have following directory tree accessible via http:

```
firmware/
        stable/
                sysupgrade/
                factory/
        snapshot/
                sysupgrade/
                factory/
        experimental/
                sysupgrade/
                factory/
```

The server must be available via IPv6.

2.2.3 Command Line

These commands can be used on a node:

```
# Update with some probability
autoupdater
```

```
# Force update check, even when the updater is disabled
autoupdater -f
```

2.3 WLAN configuration

Gluon allows to configure 2.4GHz and 5GHz radios independently. The configuration may include any or all of the three networks “client” (AP mode), “mesh” (802.11s mode) and “ibss” (adhoc mode), which can be used simultaneously (using “mesh” and “ibss” at same time should be avoided though as weaker hardware usually can’t handle the additional load). See Site configuration for details on the configuration.

2.3.1 Upgrade behaviour

For each of these networks, the site configuration may define a *disabled* flag (by default, all configured networks are enabled). This flag is merely a default setting, on upgrades the existing setting is always retained (as this setting may have been changed by the user). This means that is is not possible to enable or disable an existing network configurations during upgrades.

For the “mesh” and “ibss” networks, the default setting only has an effect if none of the two has existed before. If a new configuration has been added for “mesh” or “ibss”, while the other of the two has already existed before, the enabled/disabled state of the existing configuration will also be set for the new configuration.

This allows upgrades to change from IBSS to 11s and vice-versa while retaining the “wireless meshing is enabled/disabled” property configured by the user regardless of the used mode.

2.4 Private WLAN

It is possible to set up a private WLAN that bridges the WAN port and is seperated from the mesh network. Please note that you should not enable `mesh_on_wan` simultaneously.

The private WLAN can be enabled through the config mode if the package `gluon-luci-private-wifi` is installed. You may also enable a private WLAN using the command line:

```
uci set wireless.wan_radio0=wifi-iface
uci set wireless.wan_radio0.device=radio0
uci set wireless.wan_radio0.network=wan
uci set wireless.wan_radio0.mode=ap
uci set wireless.wan_radio0.encryption=psk2
uci set wireless.wan_radio0.ssid="$SSID"
uci set wireless.wan_radio0.key="$KEY"
uci set wireless.wan_radio0.disabled=0
uci commit
wifi
```

Please replace `$SSID` by the name of the WLAN and `$KEY` by your passphrase (8-63 characters). If you have two radios (e.g. 2.4 and 5 GHz) you need to do this for radio0 and radio1.

It may also be disabled by running:

```
uci set wireless.wan_radio0.disabled=1
uci commit
wifi
```

2.5 Wired mesh (Mesh-on-WAN/LAN)

In addition to meshing over WLAN and VPN, it is also possible to configured wired meshing over the LAN or WAN ports. This allows nodes to be connected directly or over wireless bridges.

Mesh-on-WAN can be enabled in addition to the mesh VPN, so multiple nodes in the same local network that is used as VPN uplink can also mesh directly. Enabling Mesh-on-WAN should be avoided if the local network is also bridged with a WLAN access point, as meshing over batman-adv causes large amounts of multicast traffic, which will take up a lot of airtime.

Enabling Mesh-on-LAN will replace the normal “client network” function of the LAN ports, as client network ports may never be connected (so care must be taken to always enable Mesh-on-LAN before connecting two nodes’ LAN ports).

2.5.1 Configuration

Both Mesh-on-WAN and Mesh-on-LAN can be configured on the “Network” page of the *Expert Mode* (if the package `gluon-luci-portconfig` is installed).

It is also possible to enable Mesh-on-WAN and Mesh-on-LAN by default by adding `mesh_on_wan = true` and `mesh_on_lan = true` to `site.conf`.

Commandline configuration

Mesh-on-WAN

It’s possible to enable Mesh-on-WAN like this:

```
uci set network.mesh_wan.auto=1
uci commit
```

It may be disabled by running:

```
uci set network.mesh_wan.auto=0
uci commit
```

Mesh-on-LAN

Configuring Mesh-on-LAN is a bit more complicated:

```
uci set network.mesh_lan.auto=1
for ifname in $(cat /lib/gluon/core/sysconfig/lan_ifname); do
  uci del_list network.client.ifname=$ifname
done
uci commit
```

It may be disabled by running:

```
uci set network.mesh_lan.auto=0
for ifname in $(cat /lib/gluon/core/sysconfig/lan_ifname); do
  uci add_list network.client.ifname=$ifname
done
uci commit
```

Please note that this configuration has changed in Gluon v2016.1. Using the old commands on v2016.1 will break the corresponding Expert Mode settings.

2.6 Node monitoring

Gluon is capable of announcing information about each node to the mesh and to neighbouring nodes. This allows nodes to learn each others hostname, IP addresses, location, software versions and various other information.

2.6.1 Format of collected data

Information to be announced is currently split into three categories:

nodeinfo In this category (mostly) static information is collected. If something is unlikely to change without human intervention it should be put here.

statistics This category holds fast changing data, like traffic counters, uptime, system load or the selected gateway.

neighbours *neighbours* contains information about all neighbouring nodes of all interfaces. This data can be used to determine the network topology.

All categories will have a `node_id` key. It should be used to relate data of different catagories.

2.6.2 Accessing Node Information

There are two packages responsible for distribution of the information. For one, information is distributed across the mesh using alfred. Information between neighbouring nodes is exchanged using *gluon-respondd*.

alfred (mesh bound)

The package `gluon-alfred` is required for this to work.

Using alfred both categories are distributed within the mesh. In order to retrieve the data you'll need both a local alfred daemon and alfred-json installed. Please note that at least one alfred daemon is required to run as *master*.

The following datatypes are used:

- *nodeinfo*: 158
- *statistics*: 159
- *neighbours*: 160

All data is compressed using GZip (alfred-json can handle the decompression).

In order to retrieve statistics data you could run:

```
# alfred-json -z -r 159
{
  "f8:d1:11:7e:97:dc": {
    "processes": {
      "total": 55,
      "running": 2
    },
    "idletime": 30632.290000000001,
    "uptime": 33200.07,
    "memory": {
      "free": 1660,
      "cached": 8268,
      "total": 29212,
      "buffers": 2236
    },
    "node_id": "f8d1117e97dc",
    "loadavg": 0.01
  },
  "90:f6:52:3e:b9:50": {
    "processes": {
      "total": 58,
      "running": 2
    },
    "idletime": 28047.470000000001,
    "uptime": 33307.849999999999,
    "memory": {
      "free": 2364,
      "cached": 7168,
      "total": 29212,
      "buffers": 1952
    },
    "node_id": "90f6523eb950",
    "loadavg": 0.34000000000000002
  }
}
```

You can find more information about alfred in its README.

gluon-respondd

gluon-respondd allows querying neighbouring nodes for their information. It is a daemon listening on the multicast address `ff02::2:1001` on UDP port 1001 on both the bare mesh interfaces and *br-client*. Unicast requests are supported as well.

The supported requests are:

- `nodeinfo`, `statistics`, `neighbours`: Returns the data of single category uncompressed.
- `GET nodeinfo`, ...: Returns the data of one or multiple categories (separated by spaces) compressed using the *deflate* algorithm (without a gzip header). The data may be decompressed using zlib and many zlib bindings using -15 as the window size parameter.

gluon-neighbour-info

The programm *gluon-neighbour-info* can be used to retrieve information from other nodes.

```
gluon-neighbour-info -i wlan0 \
-p 1001 -d ff02:0:0:0:0:0:2:1001 \
-r nodeinfo
```

An optional timeout may be specified, e.g. *-t 5* (default: 3 seconds). See the usage information printed by `gluon-neighbour-info -h` for more information about the supported arguments.

2.6.3 Adding a data provider

To add a provider, you need to install a shared object into `/lib/gluon/respondd`. For more information, refer to the respondd README and have a look the existing providers.

2.7 Adding SSH public keys

By using the package `gluon-authorized-keys` it is possible to add SSH public keys to an image to permit root login.

If you select this package, add a list of authorized keys to `site.conf` like this::

```
{
  authorized_keys = { 'ssh-rsa AAA.... user1@host',
                      'ssh-rsa AAA.... user2@host },
  hostname_prefix = ...
  ...
```

Existing keys in `/etc/dropbear/authorized_keys` will be preserved.

2.8 Roles

It is possible to define a set of roles you want to distinguish at backend side. One node can own one role which it will announce via alfred inside the mesh. This will make it easier to differentiate nodes when parsing alfred data. E.g to count only **normal** nodes and not the gateways or servers (nodemap). A lot of things are possible.

For this the section `roles` in `site.conf` is needed:

```
roles = {
  default = 'node',
  list = {
    node = 'Normal Node',
    test = 'Test Node',
    backbone = 'Backbone Node',
    service = 'Service Node',
  },
},
```

The value of `default` is the role every node will initially own. This value should be part of `list` as well. If you want node owners to change the defined roles via config-mode you can add the package `gluon-luci-node-role` to your `site.mk`. Then, you can select one of the defined roles from a dropdown list where the right-handed value is the one which is displayed and the left-handed key the one which is configured into the system.

The role is saved in `gluon-node-info.system.role`. To change the role using command line do:

```
uci set gluon-node-info.system.role="$ROLE"
uci commit
```

Please replace `$ROLE` by the role you want the node to own.

CHAPTER 3

Developer Documentation

3.1 Development Basics

Gluon's source is kept in git repositories at GitHub.

3.1.1 Bug Tracker

The main repo does have issues enabled.

3.1.2 IRC

Gluon's developers frequent #gluon on hackint. You're welcome to join us!

3.1.3 Working with repositories

To update the repositories used by Gluon, just adjust the commit IDs in *modules* and rerun

```
make update
```

make update also applies the patches that can be found in the directories found in *patches*; the resulting branch will be called *patched*, while the commit specified in *modules* can be refered to by the branch *base*.

```
make unpatch
```

sets the repositories to the *base* branch,

```
make patch
```

re-applies the patches by resetting the *patched* branch to *base* and calling *git am* for the patch files. Calling *make* or a similar command after calling *make unpatch* is generally not a good idea.

After new patches have been commited on top of the patched branch (or existing commits since the base commit have been edited or removed), the patch directories can be regenerated using

```
make update-patches
```

If applying a patch fails because you have changed the base commit, the repository will be reset to the old *patched* branch and you can try rebasing it onto the new *base* branch yourself and after that call *make update-patches* to fix the problem.

Always call *make update-patches* after making changes to a module repository as *make update* will overwrite your commits, making *git reflog* the only way to recover them!

3.2 Adding support for new hardware

This page will give a short overview on how to add support for new hardware to Gluon.

3.2.1 Hardware requirements

Having an ath9k (or ath10k) based WLAN adapter is highly recommended, although other chipsets may also work. VAP (multiple SSID) support is a requirement. At the moment, Gluon's scripts can't handle devices without WLAN adapters (although such environments may also be interesting, e.g. for automated testing in virtual machines).

3.2.2 Adding profiles

The vast majority of devices with ath9k WLAN uses the ar71xx target of OpenWrt. If the hardware you want to add support for is also ar71xx, adding a new profile is enough.

Profiles are defined in `targets/<target>-<subtarget>/profiles.mk`. There are two macros used to define which images are generated: `GluonProfile` and `GluonModel`. The following examples are taken from `profiles.mk` of the `ar71xx-generic` target:

```
$(eval $(call GluonProfile,TLWR1043))
$(eval $(call GluonModel,TLWR1043,tl-wr1043nd-v1-squashfs,tp-link-tl-wr1043n-nd-v1))
$(eval $(call GluonModel,TLWR1043,tl-wr1043nd-v2-squashfs,tp-link-tl-wr1043n-nd-v2))
```

The `GluonProfile` macro takes at least one parameter, the profile name as it is defined in the Makefiles of OpenWrt (`openwrt/target/linux/<target>/<subtarget>/profiles/*` and `openwrt/target/linux/<target>/image/Makefile`). If the target you are on doesn't define profiles (e.g. on x86), just add a single profile called `Generic` or similar.

It may optionally take a second parameter which defines additional packages to include for the profile (e.g. ath10k). The additional packages defined in `openwrt/target/linux/<target>/<subtarget>/profiles/*` aren't used.

The `GluonModel` macro takes three parameters: The profile name, the suffix of the image file generated by OpenWrt (without the file extension), and the final image name of the Gluon image. The final image name must be the same that is returned by the following command.

```
lua -e 'print(require("platform_info").get_image_name())'
```

This is just so the autoupdater can work. The command has to be executed _on_ the target (eg. the hardware router with a flashed image). So you'll first have to build an image with a guessed name, and afterwards build a new, correctly named image. On targets which aren't supported by the autoupdater, `require("platform_info").get_image_name()` will just return `nil` and the final image name may be defined arbitrarily.

On devices with multiple WLAN adapters, care must also be taken that the primary MAC address is configured correctly. `/lib/gluon/core/sysconfig/primary_mac` should contain the MAC address which can be found on a label on most hardware; if it does not, `/lib/gluon/upgrade/core/initial/001-sysconfig` in `gluon-core` might need a fix. (There have also been cases in which the address was incorrect even on devices with only one WLAN adapter, in these cases an OpenWrt bug was the cause).

3.2.3 Adding support for new hardware targets

Adding a new target is much more complex than adding a new profile. There are two basic steps required for adding a new target:

Adjust packages

One package that definitely needs adjustments for every new target added is `lua-platform-info`. Just start with a copy of an existing platform info script, adjust it for the new target, and add the new target to the list of supported targets in the package Makefile.

On many targets, Gluon's network setup scripts (mainly in the packages `gluon-core` and `gluon-mesh-batman-adv-core`) won't run correctly without some adjustments, so better double check that everything is fine there (and the files `primary_mac`, `lan_ifname` and `wan_ifname` in `/lib/gluon/core/sysconfig/` contain sensible values).

Add support to the build system

A directory for the new target must be created under `targets`, and it must be added to `targets/targets.mk`. In the new target directory, the following files must be created:

- profiles.mk
- config (optional)

For `profiles.mk`, see *Adding profiles*. The file `config` can be used to add additional, target-specific options to the OpenWrt config.

After this, is should be sufficient to call `make GLUON_TARGET=<target>` to build the images for the new target.

3.3 Upgrade scripts

3.3.1 Basics

After each sysupgrade (including the initial installation), Gluon will execute all scripts under `/lib/gluon/upgrade`. These scripts' filenames usually begin with a 3-digit number specifying the order of execution.

To get an overview of the ordering of all scripts of all packages, the helper script `contrib/lsupgrade.sh` in the Gluon repository can be used, which will print all upgrade scripts' filenames and directories. If executed on a TTY, the filename will be highlighted in green, the repository in blue and the package in red.

3.3.2 Best practices

- Most upgrade scripts are written in Lua. This allows using lots of helper functions provided by LuCi and Gluon, e.g. to access the site configuration or edit UCI configuration files.
- Whenever possible, scripts shouldn't check if they are running for the first time, but just edit configuration files to achive a valid configuration (without overwriting configuration changes made by the user where desirable). This allows using the same code to create the initial configuration and upgrade configurations on upgrades.

- If it is unavoidable to run different code during the initial installation, the `sysconfig.gluon_version` variable can be checked. This variable in `nil` during the initial installation and contains the previously install Gluon version otherwise. The package `gluon-legacy` (which is responsible for upgrades from the old firmwares of Hamburg/Kiel/Lübeck) uses the special value `legacy`; other packages should handle this value just as any other string.

3.3.3 Script ordering

These are some guidelines for the script numbers:

- `0xx`: Basic `sysconfig` setup
- `1xx`: Basic system setup (including basic network configuration)
- `2xx`: Wireless setup
- `3xx`: Advanced network and system setup
- `4xx`: Extended network and system setup (e.g. mesh VPN and next-node)
- `5xx`: Miscellaneous (everything not fitting into any other category)
- `6xx` .. `8xx`: Currently unused
- `9xx`: Upgrade finalization

3.4 Config Mode

As of 2014.4 *gluon-config-mode* consists of several modules.

gluon-config-mode-core This modules provides the core functionality for the config mode. All modules must depend on it.

gluon-config-mode-hostname Provides a hostname field.

gluon-config-mode-autoupdater Informs whether the autoupdater is enabled.

gluon-config-mode-mesh-vpn Allows toggling of mesh-vpn-fastd and setting a bandwidth limit.

gluon-config-mode-geo-location Enables the user to set the geographical location of the node.

gluon-config-mode-contact-info Adds a field where the user can provide contact information.

In order to get a config mode close to the one found in 2014.3.x you may add these modules to your *site.mk*: gluon-config-mode-hostname, gluon-config-mode-autoupdater, gluon-config-mode-mesh-vpn, gluon-config-mode-geo-location, gluon-config-mode-contact-info

3.4.1 Writing Config Mode Modules

Config mode modules are located at */lib/gluon/config-mode/wizard* and */lib/gluon/config-mode/reboot*. Modules are named like *0000-name.lua* and are executed in lexical order. If you take the standard set of modules, the order is, for wizard modules:

- 0050-autoupdater-info
- 0100-hostname
- 0300-mesh-vpn
- 0400-geo-location

- 0500-contact-info

While for reboot modules it is:

- 0100-mesh-vpn
- 0900-msg-reboot

Wizards

Wizard modules return a UCI section. A simple module capable of changing the hostname might look like this:

```
local cbi = require "luci.cbi"
local uci = luci.model.uci.cursor()

local M = {}

function M.section(form)
  local s = form:section(cbi.SimpleSection, nil, nil)
  local o = s:option(cbi.Value, "_hostname", "Hostname")
  o.value = uci:get_first("system", "system", "hostname")
  o.rmempty = false
  o.datatype = "hostname"
end

function M.handle(data)
  uci:set("system", uci:get_first("system", "system"), "hostname", data._hostname)
  uci:save("system")
  uci:commit("system")
end

return M
```

Reboot page

Reboot modules return a function that will be called when the page is to be rendered or nil (i.e. the module is skipped):

```
if no_hello_world_today then
  return nil
else
  return function ()
    luci.template.render_string("Hello World!")
  end
end
```

3.5 WAN support

As the WAN port of a node will be connected to a user's private network, it is essential that the node only uses the WAN when it is absolutely necessary. There are two cases in which the WAN port is used:

- Mesh VPN (package `gluon-mesh-vpn-fastd`
- DNS to resolve the VPN servers' addresses (package `gluon-wan-dnsmasq`)

After the VPN connection has been established, the node should be able to reach the mesh's DNS servers and use these for all other name resolution.

3.5.1 Routing tables

As a node may get IPv6 default routes both over the WAN and the mesh, Gluon uses two routing tables for IPv6. As all normal traffic should go over the mesh, the mesh routes are added to the default table (table 0). All routes on the WAN interface are put into table 1 (see `/lib/gluon/upgrade/110-network` in `gluon-core`).

There is also an *ip -6 rule* which routes all IPv6 traffic with a packet mark with the bit 1 set though table 1.

3.5.2 libpacketmark

libpacketmark is a library which can be loaded with `LD_PRELOAD` and will set the packet mark of all sockets created by a process in accordance with the `LIBPACKETMARK_MARK` environment variable. This allows setting the packet mark for processes which don't support this themselves. The process must run as root (or at least with `CAP_NET_ADMIN`) for this to work.

Unfortunately there's no nice way to set the packet mark via iptables for outgoing packets. The iptables will run after the packet has been created, to even when the packet mark is changed and the packet is re-routed, the source address won't be rewritten to the default source address of the newly chosen route. *libpacketmark* avoids this issue as the packet mark will already be set when the packet is created.

3.5.3 gluon-wan-dnsmasq

To separate the DNS servers in the mesh from the ones on the WAN, the `gluon-wan-dnsmasq` package provides a secondary DNS daemon which runs on `127.0.0.1:54`. It will automatically use all DNS servers explicitly configured in `/etc/config/gluon-wan-dnsmasq` or received via DNS/RA on the WAN port. It is important that no DNS servers for the WAN interface are configured in `/etc/config/network` and that `peerdns` is set to 0 so the WAN DNS servers aren't leaked to the primary DNS daemon.

libpacketmark is used to make the secondary DNS daemon send its requests over the WAN interface.

The package `gluon-mesh-vpn-fastd` provides an iptables rule which will redirect all DNS requests from processes running with the primary group `gluon-fastd` to `127.0.0.1:54`, thus making fastd use the secondary DNS daemon.

3.6 Internationalization support

3.6.1 General guidelines

- All config mode packages must be fully translatable, with complete English and German texts.
- All new expert mode packages be fully translatable. English texts are required, German texts recommended.
- Existing expert mode packages should be made translatable as soon as possible.
- The "message IDs" (which are the arguments to the `translate` function) should be the English texts.

3.6.2 i18n support in LuCI

Internationalization support can be found in the `luci.i18n` package. Strings are translated using the `i18n.translate` and `i18n.translatef` functions (`translate` for static strings, `translatef` for printf-like formatted string).

Example from the `gluon-config-mode-geo-location` package:

```
local i18n = require "luci.i18n"
o = s:option(cbi.Flag, "_location", i18n.translate("Show node on the map"))
```

3.6.3 Adding translation templates to Gluon packages

The i18n support is based on the standard gettext system. For each translatable package, a translation template with extension `.pot` can be created using the `i18n-scan.pl` script from the LuCI repository:

```
cd package/gluon-config-mode-geo-location
mkdir i18n
cd i18n
../../../packages/luci/build/i18n-scan.pl ../files > gluon-config-mode-geo-location.pot
```

The entries in the template can be reordered after the generation if desirable. Lots of standard translations like "Cancel" are already available in the LuCI base translation file (see `packages/luci/po/templates/base.pot`) and can be removed from the template.

In addition, some additions to the Makefile must be made. Instead of OpenWrt's default package.mk, the Gluon version `$(GLUONDIR)/include/package.mk` must be used. The i18n files must be installed and PKG_CONFIG_DEPENDS must be added:

```
...
include $(GLUONDIR)/include/package.mk

PKG_CONFIG_DEPENDS += $(GLUON_I18N_CONFIG)
...
define Build/Compile
  $(call GluonBuildI18N,gluon-config-mode-geo-location,i18n)
endef

define Package/gluon-config-mode-geo-location/install
  ...
  $(call GluonInstallI18N,gluon-config-mode-geo-location,$(1))
endef
...
```

3.6.4 Adding translations

A new translation file for a template can be added using the `msginit` command:

```
cd package/gluon-config-mode-geo-location/i18n
msginit -l de
```

This will create the file `de.po` in which the translations can be added.

The translation file can be updated to a new template version using the `msgmerge` command:

```
msgmerge -U de.po gluon-config-mode-geo-location.pot
```

After the merge, the translation file should be checked for "fuzzy matched" entries where the original English texts have changed. All entries from the the translation file should be translated in the `.po` file (or removed from it, so the original English texts are displayed instead).

3.6.5 Adding support for new languages

A list of all languages supported by LuCI can be found in the `packages/luci/luci.mk` file after Gluon's dependencies have been downloaded using `make update`. Adding translations for these languages is straightforward using the `msginit` command.

For other languages, support must be added tu LuCI first, which constitutes completely translating the `base.pot`. Please contact the upstream LuCI maintainers if you'd like to do this.

CHAPTER 4

Packages

4.1 gluon-client-bridge

This package provides a bridge (*br-client*) for connecting clients. It will also setup a wireless interface, provided it is configured in *site.conf*.

4.1.1 site.conf

wifi24.ap.ssid / wifi5.ap.ssid SSID for the client network

4.2 gluon-ebtables-filter-multicast

The *gluon-ebtables-filter-multicast* package filters out various kinds of non-essential multicast traffic, as this traffic often constitutes a disproportionate burden on the mesh network. Unfortunately, this breaks many useful services (Avahi, Bonjour chat, ...), but this seems unavoidable, as the current Avahi implementation is optimized for small local networks and causes too much traffic in large mesh networks.

The multicast packets are filtered between the nodes' client bridge (*br-client*) and mesh interface (*bat0*) on output.

The following packet types are considered essential and aren't filtered:

- ARP (except requests for/replies from 0.0.0.0)
- DHCP, DHCPv6
- ICMPv6 (except Echo Requests (ping) and Node Information Queries (RFC4620)
- IGMP

In addition, the following packet types are allowed to allow experimentation with layer 3 routing protocols.

- Babel
- OSPF
- RIPng

The following packet types are also allowed:

- BitTorrent Local Peer Discovery (it seems better to have local peers for BitTorrent than sending everything through the internet)

4.3 gluon-ebtables-filter-ra-dhcp

The *gluon-ebtables-filter-ra-dhcp* package tries to prevent common misconfigurations (i.e. connecting the client interface of a Gluon node to a private network) from causing issues for either of the networks.

The rules are the following:

- DHCP requests, DHCPv6 requests and Router Solicitations may only be sent from clients to the mesh, but aren't forwarded from the mesh to clients
- DHCP replies, DHCPv6 replies and Router Advertisements from clients aren't forwarded to the mesh

CHAPTER 5

Releases

5.1 Gluon 2016.1.3

5.1.1 Added hardware support

ar71xx-generic

- ALFA Hornet UB / AP121 / AP121U
- TP-Link TL-WA7510N

5.1.2 Bugfixes

- The nondeterministic boot hang (#669) that was thought to be fixed in Gluon v2016.1.2 has resurfaced on other hardware. We believe it is now fixed properly.
- Sysupgrades on the Xen DomU have been fixed.
- Gluon can now be built on systems that use LibreSSL instead of OpenSSL.

5.1.3 Known Issues

- Default TX power on many Ubiquiti devices is too high, correct offsets are unknown (#94)

  Reducing the TX power in the Expert Mode is recommended.
- The MAC address of the WAN interface is modified even when Mesh-on-WAN is disabled (#496)

  This may lead to issues in environments where a fixed MAC address is expected (like VMware when promicious mode is disallowed).
- Inconsistent respondd API (#522)

  The current API is inconsistent and will be replaced eventually. The old API will still be supported for a while.
- Unwritable flash on some Ubiquiti PicoStations (#687)

  Gluon v2016.1.1 added support for Ubiquiti AirMAX devices with AirOS 5.6.x without downgrading AirOS first before flashing Gluon. It was discovered that on Ubiquiti PicoStations, this downgrade is still necessary, as the flash is not correctly unlocked, leaving the device unable to leave Config Mode and making regular sysupgrades impossible.

  TFTP recovery can be used in this state to flash a new firmware.

5.2 Gluon 2016.1.2

5.2.1 Added hardware support

The *x86-generic* images now contain the ATIIXP PATA driver, adding support for FUTRO Thin Clients.

5.2.2 Bugfixes

A nondeterministic boot hang (#669) has been fixed. The TL-WR841N v5 seems to be affected in particular, but the kernel bug is not hardware-specific per se.

5.2.3 Known Issues

- Default TX power on many Ubiquiti devices is too high, correct offsets are unknown (#94)

  Reducing the TX power in the Expert Mode is recommended.
- The MAC address of the WAN interface is modified even when Mesh-on-WAN is disabled (#496)

  This may lead to issues in environments where a fixed MAC address is expected (like VMware when promicious mode is disallowed).
- Inconsistent respondd API (#522)

  The current API is inconsistent and will be replaced eventually. The old API will still be supported for a while.

5.3 Gluon 2016.1.1

5.3.1 Added hardware support

ar71xx-generic

- Onion Omega
- TP-Link TL-MR13U v1

5.3.2 Bugfixes

Build

Don't overwrite the opkg repository key on each build.

AirOS 5.6.x compatiblity

Downgrading to AirOS 5.5.x before flashing Gluon on Airmax M XM/XW devices (NanoStation, Bullet, ...) is not necessary anymore.

Status page

- Fix purging of disappered neighbours from the list
- Don't clear the signal graphs when scrolling in mobile browsers
- Improve browser compability (don't assume the Internationalization API is available, fixes the display of numbers in Firefox for Android)

Config mode

- Strip trailing whitespace from number input fields (LuCI's validator doesn't catch this)
- Don't allow negative bandwidth limits

Failsafe mode

- Fix entering the failsafe mode on the TL-WDR4900.

5.3.3 Known Issues

- Default TX power on many Ubiquiti devices is too high, correct offsets are unknown (#94)

  Reducing the TX power in the Expert Mode is recommended.
- The MAC address of the WAN interface is modified even when Mesh-on-WAN is disabled (#496)

  This may lead to issues in environments where a fixed MAC address is expected (like VMware when promicious mode is disallowed).
- Inconsistent respondd/announced API (#522)

  The current API is inconsistent and will be replaced eventually. The old API will still be supported for a while.
- Nondeterministic production of broken images for some (very old) hardware (#669)

  At the moment it seems like only the TL-WR841N v5 is affected.

5.4 Gluon 2016.1

5.4.1 Added hardware support

ar71xx-generic

- Buffalo
  - WZR-HP-G300NH
- D-Link
  - DIR-505 (A1)
- TP-Link
  - CPE210/220/510/520 v1.1
  - TL-WA901N/ND v1

  - TL-WR710N v2
  - TL-WR801N/ND v1, v2
  - TL-WR841N/ND v10
  - TL-WR843N/ND v1
  - TL-WR940N v1, v2, v3
  - TL-WR941ND v6
  - TL-WR1043N/ND v3
- Ubiquiti
  - airGateway
  - airRouter
  - UniFi AP Outdoor+
- Western Digital
  - My Net N600
  - My Net N750

x86-xen_domu

New target containing the necessary drivers for use in Xen.

x86-64

64bit version of *x86-generic*. The generic image can also be used in KVM with VirtIO.

5.4.2 New features

Kernel module opkg repository

We've not been able to keep ABI compatiblity with the kernel of the official OpenWrt images. Therefore, Gluon now generates an opkg repository with modules itself.

The repository can be found at *output/modules/* by default, the image output directory has been moved from *images/* to *output/images/*. See the updated Getting Started guide for information on the handling of the signing keys for this repository.

The *opkg_repo* site.conf option has been replaced to allow specifying this and other additional repositories.

New status page

The new status page provides a visually pleasing experience, and displays all important information on a node in a clear manner. It also contains a real-time signal strength graph for all neighbouring nodes to aid with the alignment of antennas.

802.11s mesh support

Gluon now supports using 802.11s for its mesh links instead of IBSS (Adhoc). This will allow supporting more WLAN hardware in the future (like Ralink/Mediatek, which don't support AP and IBSS mode simultaneously).

Note that batman-adv is still used on top of 802.11s (and 802.11s forwarding is disabled), the mesh routing protocol provided by 802.11s is not used.

Multicast filter extension

The *gluon-ebtables-filter-multicast* package has been extended to filter out multicast ICMP and ICMPv6 Echo Requests (ping) and Node Information Queries (RFC4620). This prevents pings to multicast addresses like ff02::1 to cause traffic peaks (as all nodes and clients would answer such a ping).

French translation

A French translation for the Config Mode/Expert Mode has been added.

5.4.3 Bugfixes

- Update kernel code for the QCA953x

  Might improve stability of the TP-Link TL-WR841N/ND v9.
- Fix model detection on some Netgear WNDR3700v2

  The broken devices will identify as "NETGEAR ". This also breaks the autoupdater, making a manual upgrade necessary.
- Ensure that *odhcp6c* doesn't spawn multiple instances of `dhcpv6.script`
- Fix support for Buffalo WZR-600DHP

  A flashable factory image is generated now. The sysupgrade image is still shared with the WZR-HP-AG300H.

5.4.4 Site changes

- `site.conf`
  - New WLAN configuration

    `wifi24` and `wifi5` need to be updated to a new more flexible format. A configuration using the old format

```
{
  channel = 1,
  htmode = 'HT20'
  ssid = 'entenhausen.freifunk.net',
  mesh_ssid = 'xe:xx:xx:xx:xx:xx',
  mesh_bssid = 'xe:xx:xx:xx:xx:xx',
  mesh_mcast_rate = 12000,
}
```

    would look like this in the new format:

```
{
  channel = 1,
  ap = {
    ssid = 'entenhausen.freifunk.net',
  },
  ibss = {
    ssid = 'xe:xx:xx:xx:xx:xx',
    bssid = 'xe:xx:xx:xx:xx:xx',
    mcast_rate = 12000,
  },
}
```

The `htmode` option has been dropped, the channel width is now always set to 20MHz (see https://github.com/freifunk-gluon/gluon/issues/487 for a discussion of this change).

In addition to the old IBSS (Adhoc) based meshing, 802.11s-based meshing can be configured using the `mesh` section. Example:

```
{
  channel = 1,
  ap = {
    ssid = 'entenhausen.freifunk.net',
  },
  mesh = {
    id = 'mesh.entenhausen.freifunk.net', -- can by any string, human-readable or random
    mcast_rate = 12000,
  },
}
```

While using `ibss` and `mesh` at the same time is possible, is causes high load in very active meshes, so it is advisable to avoid such configurations.

– Bandwidth limitation defaults

The old section `simple_tc.mesh_vpn` has been moved to `fastd_mesh_vpn.bandwidth_limit` and the `ifname` field isn't used anymore. What looked like this before

```
simple_tc = {
  mesh_vpn = {
    ifname = 'mesh-vpn',
    enabled = false,
    limit_ingress = 3000,
    limit_egress = 200,
  }
}
```

needs to be changed to

```
fastd_mesh_vpn = {
  -- ...

  bandwidth_limit = {
    enabled = false,
    ingress = 3000,
    egress = 200,
  },
}
```

– opkg repository configuration

The opkg configuration has been changed to be more flexible and allow specifying custom repositories. Example:

```
opkg = {
  openwrt = 'http://opkg.services.ffeh/openwrt/%n/%v/%S/packages',
  extra = {
    modules = 'http://opkg.services.ffeh/modules/gluon-%GS-%GR/%S',
  },
}
```

The keys of the `extra` table (like `modules` in this example) can be chosen arbitrarily.

Instead of explicitly specifying the whole URL, using patterns is recommended. The following patterns are understood:

- `%n` is replaced by the OpenWrt version codename (e.g. "chaos_calmer")
- `%v` is replaced by the OpenWrt version number (e.g. "15.05")
- `%S` is replaced by the target architecture (e.g. "ar71xx/generic")
- `%GS` is replaced by the Gluon site code (as specified in `site.conf`)
- `%GV` is replaced by the Gluon version
- `%GR` is replaced by the Gluon release (as specified in `site.mk`)

- `site.mk`
  - The packages *gluon-announce* and *gluon-announced* were merged into the package *gluon-respondd*. If you had any of them (probably *gluon-announced*) in your package list, you have to replace them.
- `i18n/`
  - The translations of `gluon-config-mode:pubkey` now have to show the fastd public key themselves if desired, making the formatting of the key and whether it is shown at all configurable. To retain the old format, add <p> to the beginning of your translations and append:

```
"</p>"
"<div class=\"the-key\">"
" # <%= hostname %>"
" <br/>"
"<%= pubkey %>"
"</div>"
```

5.4.5 Internals

- OpenWrt has been updated to Chaos Calmer
- mac80211 has been backported from OpenWrt trunk r47249 (wireless-testing 2015-07-21)

  This allows us to support the TL-WR940N v3/TL-WR941ND v6, which uses a TP9343 (QCA956x) SoC.
- Several packages have been moved from the Gluon repo to the packages repo, removing references to Gluon:
  - gluon-cron -> micrond (the crontabs are now read from `/usr/lib/micron.d` instead of `/lib/gluon/cron`)
  - gluon-radvd -> uradvd
  - gluon-simple-tc -> simple-tc (the config file has been renamed as well)
- Some of the Gluon-specific i18n support code in the build system has been removed, as LuCI now provides similar facilities

- The C-based *luci-lib-jsonc* library is now used for JSON encoding/decoding instead of the pure Lua *luci-lib-json*
- The site config is now stored as JSON on the node. The Lua interface `gluon.site_config` is still available, and a C interface was added as part of the new package *libgluonutil*.
- The *respondd* daemon now uses C modules instead of Lua snippets, which greatly enhances response speed and reduces memory usage. The Gluon integration package has been renamed from *gluon-announced* to *gluon-respondd*.

5.4.6 Known Issues

- Default TX power on many Ubiquiti devices is too high, correct offsets are unknown (#94)

  Reducing the TX power in the Expert Mode is recommended.
- batman-adv causes stability issues for both alfred and respondd/announced (#177)
- The MAC address of the WAN interface is modified even when Mesh-on-WAN is disabled (#496)

  This may lead to issues in environments where a fixed MAC address is expected (like VMware when promicious mode is disallowed).
- Inconsistent respondd/announced API (#522)

  The current API is inconsistent and will be replaced in the next release. The old API will still be supported for a while.

5.5 Gluon 2015.1.2

5.5.1 Added hardware support

ar71xx-generic

- TP-Link
  - TL-WA701N/ND (v2)
  - TL-WA801N/ND (v1)
  - TL-WA830RE (v2)
  - TL-WR740N / TL-WR741ND (v5)

5.5.2 New features

- Ubiquiti Loco M, Picostation M and Rocket M now get their own images (which are just copies of the Bullet M image) so it's more obvious for users which image to use
- The x86-generic images now contain the e1000e ethernet driver by default

5.5.3 Bugfixes

- Fix download of OpenSSL during build because of broken OpenSSL download servers (again...)
- Fix another ABI incompatiblity with the upstream kernel modules which prevented loading some filesystem-related modules

- Fix potential MAC address conflicts on x86 target when using mesh-on-wan/lan
- Fix signal strength indicators on TP-LINK CPE210/510
- Fix the model name string on some NETGEAR WNDR3700v2
- Fix 5GHz WLAN switching channels and losing connectivity when other WLANs using the same channel are detected (including other Gluon nodes...); see https://github.com/freifunk-gluon/gluon/issues/386
- Fix DNS resolution for mesh VPN on IPv6-only WAN; see https://github.com/freifunk-gluon/gluon/issues/397
- gluon-mesh-batman-adv-15: update batman-adv to 2015.0 with additional bugfixes (fixes various minor bugs)
- gluon-mesh-batman-adv-15: fix forwarding of fragmented frames over multiple links with different MTUs

  batman-adv compat 15 doesn't re-fragment frames that are fragmented already. In particular, this breaks transmission of large packets which are first fragmented for mesh-on-lan/wan and are then sent over the mesh VPN, which has an even smaller MTU. Work around this limitation by decreasing the maximum fragment size to 1280, so they can always be forwarded as long there's no link with a MTU smaller than 1280.

  See https://github.com/freifunk-gluon/gluon/issues/435

5.6 Gluon 2015.1.1

5.6.1 Added hardware support

ar71xx-generic

- TP-Link
  - TL-WA830RE (v1)

5.6.2 New features

The *x86-generic* and *x86-kvm_guest* images now support two ethernet interfaces by default. If two interfaces exist during the first boot, *eth0* will be used as LAN and *eth1* as WAN.

5.6.3 Bugfixes

- Fix German "Expert Mode" label (was "Export Mode")
- Fix download of OpenSSL during build (because of broken OpenSSL download servers...)
- Fix ABI break causing kernel panics when trying to use network-related modules from the official OpenWrt repository (like *kmod-pppoe*)
- Fix race conditions breaking parallel build occasionally
- A broken network configuration would be generated when an older Gluon version was updated to 2015.1 with `mesh_on_lan` enabled in *site.conf*
- Minor announced/alfred JSON format fixes (don't output empty lists where empty objects would be expected)

5.7 Gluon 2015.1

5.7.1 Added hardware support

Gluon v2015.1 is the first release to officially support hardware that is not handled by the *ar71xx-generic* OpenWrt target. This also means that *ar71xx-generic* isn't the default target anymore, the `GLUON_TARGET` variable must be set for all runs of `make` and `make clean` now.

ar71xx-generic

- Allnet
  - ALL0315N
- D-Link
  - DIR-615 (C1)
- GL-Inet
  - 6408A (v1)
  - 6416A (v1)
  - WRT160NL
- Netgear
  - WNDR3700 (v1, v2)
  - WNDR3800
  - WNDRMAC (v2)
- TP-Link
  - TL-MR3220 (v2)
  - TL-WA701N/ND (v1)
  - TL-WA860RE (v1)
  - TL-WA901N/ND (v2, v3)
  - TL-WR743N/ND (v1, v2)
  - TL-WR941N/ND (v5)
  - TL-WR2543N/ND (v1)
- Ubiquiti
  - Nanostation M XW
  - Loco M XW
  - UniFi AP Pro

ar71xx-nand

- Netgear
  - WNDR3700 (v4)

  - WNDR4300 (v1)

mpc85xx-generic

- TP-Link
  - TL-WDR4900 (v1)

x86-generic

- x86-generic
- x86-virtualbox
- x86-vmware

x86-kvm_guest

- x86-kvm

5.7.2 New features

Multilingual config mode

All config and expert mode modules contain both English and German texts now. The English locale should always be enabled in `site.mk` (as English is the fallback language), German can be enabled in addition using the `GLUON_LANGS` setting.

The language shown is autmatically determined from the headers sent by the user's browser.

Mesh-on-LAN

Gluon now supports meshing using a node's LAN ports. It can be enabled by default in *site.conf*, and configured by the user using the *gluon-luci-portconfig* expert mode package.

Please note that nodes without the *mesh-on-lan* feature enabled must never be connected via their LAN ports.

Extended WLAN configuration

The new `client_disabled` and `mesh_disabled` keys in the `wifi24` and `wifi5` sections allow to disable the client and mesh networks by default, which may make sense for images for special installations.

The new package *gluon-luci-wifi-config* allows the user to change these settings; in addition, the WLAN adapters' transmission power can be changed in this package.

fastd "performance mode"

The new package *gluon-luci-mesh-vpn-fastd* allows the user to switch between the *security* and *performance* VPN settions. In *performance mode*, the method *null* will be prepended to the method list.

The new option `configurable` in the `fastd_mesh_vpn` section of `site.conf` must be set to *true* so firmware upgrades don't overwrite the method list completely (non-*null* methods will still be overwritten). Adding the *gluon-luci-mesh-vpn-fastd* package enforces this setting.

Altitude setting in *gluon-config-mode-geo-location*

The *gluon-config-mode-geo-location* config mode module now contains an optional altitude field.

gluon-announced rework

The *gluon-announced* package has been reworked to allow querying it from anywhere in the mesh. In contrast to *gluon-alfred*, it is based on a query-response model (the master multicasts a query, the nodes respond), while *gluon-alfred* uses periodic announcements.

For now, we recommend including both *gluon-alfred* and *gluon-announced* in Gluon-based firmwares, until *gluon-announced* is ready to replace *gluon-alfred* completely, and software like the ffmap backend has been adjusted accordingly.

Nested peer groups

Nested peer groups for the *fastd-mesh-vpn-fastd* package can now be configured in `site.conf`, each with its own peer limit. This allows to add additional constaints, for example to connect to 2 peers altogether, but only 1 peer in each data center.

Autoupdater manual branch override

When running the updater manually on the command line, the branch to use can now be overridden using the `-b` option.

5.7.3 Bugfixes

Accidental factory reset fix

Pressing a node's reset button for more than 5 seconds would completely reset a node's configuration under certain conditions.

WAN IPv6 issues

The WAN port would stop to respond to IPv6 packets sometimes, also breaking IPv6 VPN connectivity.

WDR4900 WAN MAC address

The MAC address on the WAN port of the WDR4900 was broken, making this device unusable for *mesh-on-wan* configurations.

5.7.4 Site changes

- `site.conf`
  - `hostname_prefix` is now optional, and is concatenated directly with the generated node ID, in particular no hyphen is inserted anymore. If you want to keep the old behaviour, you have to append the hyphen to the `hostname_prefix` field of your `site.conf`.

  - mesh_vpn_fastd: The default peer group name backbone isn't hardcoded anymore, any group name can be used. Instead, the fastd_mesh_vpn table must now contain an element groups, for example:

```
fastd_mesh_vpn = {
    methods = {'salsa2012+umac'},
    mtu = 1426,
    groups = {
        backbone = {
            limit = 2,
            peers = {
                -- ...
            }
        }
    }
}
```

  - config_mode: The config mode messages aren't configured in site.conf anymore. Instead, they are defined language-specific gettext files in the i18n subdirectory of the site configuration (see *Config mode texts*).
  - roles: The display strings for the node roles aren't configured in the site.conf anymore, but in the site i18n files. The site.conf section becomes:

```
roles = {
    default = 'foo',
    list = {
        'foo',
        'bar',
    }
}
```

    The display string use i18n message IDs like gluon-luci-node-role:role:foo and gluon-luci-node-role:role:bar.

- site.mk
  - gluon-mesh-batman-adv-15 is now the recommended batman-adv version for new Gluon deployments.
  - The packages gluon-setup-mode and gluon-config-mode-core must now be added to GLUON_SITE_PACKAGES explicitly (to allow replacing them with community-specific implementations).
  - The new GLUON_LANGS variable selects the config mode languages to include. It defaults to en, setting it to en de will select both the English and German locales. en must always be included.

5.7.5 Internals

New upgrade script directory

The distinction between *initial* and *invariant* scripts has been removed, all scripts are now run on each upgrade. Instead of having one script directory per package, all upgrade scripts lie in /lib/gluon/upgrade now, so it is possible to define the run order across packages.

Merged package repository

The Gluon-specific packages have been moved to the package directory of the Gluon main repository. The packages repository now only contains packages that will be submitted to the OpenWrt upstream eventually.

5.7.6 Known Issues

Alfred/respondd crashes

https://github.com/freifunk-gluon/gluon/issues/177

Occasional alfred crashes may still occur. As this is caused by a kernel issue, we suspect that respondd, which gluon-announced is based on, is affected in the same way.

Ignored TX power offset on Ubiquiti AirMax devices

https://github.com/freifunk-gluon/gluon/issues/94

The default transmission power setting on many of these devices is too high. It may be necessary to make manual adjustments, for example using the `gluon-luci-wifi-config` package. The values shown by `gluon-luci-wifi-config` generally include the TX power offset (amplifier and antenna gain) where available, but on many devices the offset is inaccurate or unavailable.

5.8 Gluon 2014.4

5.8.1 Added (and removed) hardware support

- Buffalo
  - WZR-HP-AG300H / WZR-600DHP
  - WZR-HP-G450H
- D-Link
  - DIR-615 (E1) support had to be dropped
- TP-LINK
  - CPE210/220/510/520 (v1)
  - TL-MR3040 (v2
  - TL-WA750RE (v1)
  - TL-WA801N/ND (v2)
  - TL-WA850RE (v1)
  - TL-WR703N (v1)
  - TL-WR710N (v1)
  - TL-WR1043N/ND (v2)

5.8.2 New features

OpenWrt Barrier Breaker

Switching to the new OpenWrt release 14.09 ("Barrier Breaker") has yielded lots of updates for both the kernel and most packages. Besides better performance, this has also greatly improved stability (far less out-of-memory issues!).

Modular config mode

The old `gluon-config-mode` package has been split into five small packages, each providing a single section of the config mode form. This simplifies removing or replacing parts of the wizard.

See the *Site changes* section for details.

Experimental support for batman-adv compat 15

As batman-adv has broken compatiblity starting with batman-adv 2014.0 (bumping the "compat level" to 15), Gluon users must decide which batman-adv version to use. The package for the old batman-adv version `gluon-mesh-batman-adv` has been renamed to `gluon-mesh-batman-adv-14`, the new version can be used with `gluon-mesh-batman-adv-15`.

Please note that batman-adv compat 15 still isn't tested very well (and there are known bugs in the current release 2014.3), so for now we still recommend using compat 14 in "production" environments.

fastd v16

Besides other new features and bugfixes, fastd v16 support the new authentication method "UMAC". We recommend switching from the old `salsa2012+gmac` and `null+salsa2012+gmac` methods to the new `salsa2012+umac` and `null+salsa2012+umac` as UMAC is much faster and even more secure than GMAC.

Private WLAN

The new package `gluon-luci-private-wifi` allows to configure a private WLAN with WPA-PSK in the expert mode which is bridged with the WAN uplink.

Embedding SSH keys

Using `gluon-authorized-keys` it is possible to embed predefined SSH public keys to firmware images. If `gluon-config-mode-*` is left out images will be ready to mesh after the first boot with SSH running for further configuration.

Status page resolves nodenames

The tools `gluon-announced` and `gluon-neighbour-info` are now available. Using them enables the status page to resolve hostnames and IPs of a nodes' neighbours.

This will also work on devices with multiple wireless interfaces.

5.8.3 Bugfixes

- Expert Mode: Fixed all SSH keys being removed when a password was set
- `gluon-mesh-vpn-fastd`: Fixed VPN peers removed from the `site.conf` not being removed from `/etc/config/fastd`
- TL-LINK TL-WDR3600/4300: Added workaround for reboot issues
- Improved stability (due to switch to OpenWrt Barrier Breaker)

5.8.4 Site changes

- `site.mk`
  - Obsolete packages:
    - `gluon-config-mode`
    - `gluon-mesh-batman-adv`
  - Recommended new packages:
    - `gluon-config-mode-autoupdater`
    - `gluon-config-mode-hostname`
    - `gluon-config-mode-mesh-vpn`
    - `gluon-config-mode-geo-location`
    - `gluon-config-mode-contact-info`
    - `gluon-mesh-batman-adv-14` (specify this before all other packages in the `site.mk`!)

5.8.5 Internals

The switch to Barrier Breaker has led to a multitude of changes all over Gluon:

- The config mode/setup mode is now started by an own set of init scripts in `/lib/gluon/setup-mode/rc.d` run by procd
- Many tools and services used by Gluon have been replaced by our own implementations to reduce the size of the images:
  - ethtool has been replaced by our minimal Lua library *lua-ethtool-stats*
  - tc has been replaced by our minimal implementation *gluon-simple-tc*
  - radvd has been replaced by our minimal implementation *gluon-radvd*

5.8.6 Known Issues

Alfred crashes

https://github.com/freifunk-gluon/gluon/issues/177

Alfred may still crash unconditionally. Some measures have been taken to aid but the core problem hasn't been analyzed yet.

Out of memory / batman-adv memory leaks

https://github.com/freifunk-gluon/gluon/issues/216

In some (hopefully rare!) cases batman-adv may still leak memory associated with global TT entries. This may result in kernel panics or out-of-memory conditions.

Ignored tx-power offset on Ubiquiti AirMax devices

https://github.com/freifunk-gluon/gluon/issues/94

There is still no OpenWRT support for determining the transmission power offsets on Ubiquiti AirMax devices (Bullet M2, Picostation M2, Nanostation (loco) M2, ...). Use Gluon with caution on these devices! Manual adjustment may be required.

5.9 Gluon 2014.3.1

This is a bugfix release.

5.9.1 Bugfixes

- gluon-announced zombie process bug

  gluon-announced was creating zombie processes when answering requests, causing issues with the new status page which is currently in development.
- fastd peers removed from `site.conf` weren't removed correctly from the fastd configuration on firmware upgrades
- Expert Mode: setting a password will not remove SSH keys anymore
- alfred has been updated to 2014.3.0

  We hope this solves the alfred stability issues noted by several people.
- `gluon-ebtables-filter-ra-dhcp` and `gluon-ebtables-filter-multicast` have been fixed to allow DHCPv6 to work
- Another ath9k patch has been added, which might further improve WLAN stability and performance

5.9.2 New features

- Support for static WAN setups instead of (DHCP/Router Advertisement) has been added; configuration is possible on the port config page of the Expert Mode.

5.9.3 Site changes

- `site.conf`
  - The new boolean option `fastd_mesh_vpn.enabled` allows enabling the mesh VPN by default. This value is optional; if it isn't specified, the mesh VPN will be disabled.

5.10 Gluon 2014.3

5.10.1 New hardware support

- Linksys WRT160NL

5.10.2 New features

New autoupdater

The autoupdater has been rewritten.

Two new fields have been added to the manifest:

DATE Specifies the time and date the update was released. `make manifest` will take care of setting it to the correct value.

PRIORITY Specifies the maximum number of days until the update should be attempted (thus lower numbers mean the priority is higher). It must be set either in `site.mk` or on the `make manifest` command line.

Updates will be attempted at night, between 04:00 and 5:00, with a specific probability. When less than `PRIORITY` days have passed (calculated using `DATE` and the current time), the probability will proportional to the time passed. I.e. the update probability will start at 0 and slowly increase to 1 until `PRIORITY` days have passed. From then, the probability will be fixed at 1.

Note: For the new update logic to work, a valid NTP server reachable over the mesh (using IPv6) must be configured in `site.conf`. If the autoupdater is unable to determine the correct time, it will fall back to a behavior similar to the old implementation (i.e. hourly update attempts).

Seperation of announced data

The data announced by alfred has been split into two data types:

- *nodeinfo* (type 158) contains all static information about a node
- *statistics* (type 159) contains all dynamic information about a node

Both types also contain a new field `node_id` which contains an arbitrary unique ID (currently the primary MAC address, sans colons) which can be used to match the *nodeinfo* with *statistics* information.

gluon-announced

A new daemon has been added in a new package `gluon-announced`. This daemon can be used for querying the *nodeinfo* data of a node via link-local multicast on the ad-hoc interfaces.

At the moment, this daemon is not used, but we recommend including it in `site.mk` nevertheless as we plan to implement a new status page showing some information about neighbor nodes in the next version of Gluon.

VPN over IPv6

It is now possible to use fastd in IPv6 WAN networks. This still needs testing, but it should work well.

Please note that the MTU of 1426 used by many communities for VPN over IPv4 is too big for IPv6 as the IPv6 header is 20 bytes longer (fastd over IPv4 has an overhead of 66 bytes, fastd over IPv6 has an overhead of 86 bytes).

More modular Config Mode

The package `gluon-config-mode` has been split into multiple packages to simplify the development of extensions. The low-level logic (handling of the button, starting the services for the config mode) has been moved into a new package `gluon-setup-mode`, while `gluon-config-mode` only contains the frontend now.

Extended Expert Mode

The Expert Mode now has a nice info page. In addition, the new package `gluon-luci-portconfig` has been added which allows simple configuration of batman-adv on the WAN interface.

Site validators

The content of the `site.conf` is now validated when the images are built to make it less likely to accidentially build broken images.

gluon-firewall

The package `gluon-firewall` has been removed. Its features are now part of the packages `gluon-core` and `gluon-mesh-batman-adv`.

gluon-ath9k-workaround

This package installs a cron job which tries to recognize ath9k hangs and restart the WLAN while recording some information. It is very rudimentary and we can't really recommend using it on "production" nodes.

5.10.3 Bugfixes

Improved ath9k stability

Multiple bugs in the WLAN driver ath9k have been fixed upstream. This should greatly improve the WLAN stability.

odhcp6c 50 day bug

An important update for odhcp6c fixes a bug which caused Gluon nodes to lose their IPv6 addresses on br-client after an uptime of 50 days, making the nodes unable perform automated updates (besides other issues).

IPv6 preference

Commands like `wget` now prefer IPv6 for domains with both AAAA and A records, allowing to use such domains for the autoupdater URLs and as NTP servers in `site.conf`.

5.10.4 Site changes

- `site.conf`
  - The `probability` fields for the autoupdater branches can be dropped as they aren't used anymore
  - The type of the `enabled` options of the `gluon-simple-tc` configuration has been changed to boolean, so `true` and `false` must be used instead of 1 and 0 now
- `site.mk`
  - Obsolete packages:
    - `gluon-firewall`
  - Recommended new packages:

    * `gluon-announced`
    * `gluon-luci-portconfig`

  - GLUON_PRIORITY must be set in `site.mk` or on the `make manifest` commandline. Use `GLUON_PRIORITY ?= 0` in `site.mk` to allow overriding from the commandline.

5.10.5 Internals

Some internal changes not mentioned before which are interesting for developers:

- Many more shell scripts have been converted to Lua
- `gluon-mesh-vpn-fastd` now uses the new package `gluon-wan-dnsmasq`, which provides a secondary DNS server on port 54 that is only reachable from *localhost* and uses the DNS servers on the WAN interface for everything. This allowed us to remove some ugly hacks which were making the DNS servers used depend on the domain being resolved.

  For IPv6, the default route is now controlled via packet marks, so the secondary DNS server and fastd set the packet mark so they use the default route provided on the WAN interface instead of the mesh.

CHAPTER 6

Supported Devices & Architectures

6.1 ar71xx-generic

- ALFA Network
  - AP121
  - AP121U
  - Hornet-UB
- Allnet
  - ALL0315N
- Buffalo
  - WZR-HP-AG300H / WZR-600DHP
  - WZR-HP-G300NH
  - WZR-HP-G450H
- D-Link
  - DIR-505 (A1)
  - DIR-615 (C1)
  - DIR-825 (B1)
- GL-Inet
  - 6408A (v1)
  - 6416A (v1)
- Linksys
  - WRT160NL
- Netgear
  - WNDR3700 (v1, v2)
  - WNDR3800
  - WNDRMAC (v2)
- Onion

  - Omega
- TP-Link
  - CPE210 (v1.0, v1.1)
  - CPE220 (v1.0, v1.1)
  - CPE510 (v1.0, v1.1)
  - CPE520 (v1.0, v1.1)
  - TL-MR13U (v1)
  - TL-MR3020 (v1)
  - TL-MR3040 (v1, v2)
  - TL-MR3220 (v1, v2)
  - TL-MR3420 (v1, v2)
  - TL-WA701N/ND (v1, v2)
  - TL-WA750RE (v1)
  - TL-WA7510N (v1)
  - TL-WA801N/ND (v1, v2)
  - TL-WA830RE (v1, v2)
  - TL-WA850RE (v1)
  - TL-WA860RE (v1)
  - TL-WA901N/ND (v1, v2, v3)
  - TL-WDR3500 (v1)
  - TL-WDR3600 (v1)
  - TL-WDR4300 (v1)
  - TL-WR703N (v1)
  - TL-WR710N (v1, v2)
  - TL-WR740N (v1, v3, v4, v5)
  - TL-WR741N/ND (v1, v2, v4, v5)
  - TL-WR743N/ND (v1, v2)
  - TL-WR801N/ND (v1, v2)
  - TL-WR841N/ND (v3, v5, v7, v8, v9, v10)
  - TL-WR842N/ND (v1, v2)
  - TL-WR843N/ND (v1)
  - TL-WR940N (v1, v2, v3)
  - TL-WR941ND (v2, v3, v4, v5, v6)
  - TL-WR1043N/ND (v1, v2, v3)
  - TL-WR2543N/ND (v1)
- Ubiquiti

  - Air Gateway
  - Air Router
  - Bullet M
  - Nanostation M
  - Nanostation M XW
  - Loco M XW
  - Picostation M
  - Rocket M
  - UniFi AP
  - UniFi AP Pro
  - UniFi AP Outdoor
  - UniFi AP Outdoor+
- Western Digital
  - My Net N600
  - My Net N750

6.2 ar71xx-nand

- Netgear
  - WNDR3700 (v4)
  - WNDR4300 (v1)

6.3 mpc85xx-generic

- TP-Link
  - TL-WDR4900 (v1)

6.4 x86-generic

- x86-generic
- x86-virtualbox
- x86-vmware

See also: x86 support

6.5 x86-kvm_guest

- x86-kvm

See also: x86 support

6.6 x86-xen_domu

- x86-xen

See also: x86 support

6.7 x86-64

- x86-64-generic
- x86-64-virtualbox
- x86-64-vmware

See also: x86 support

CHAPTER 7

License

See LICENCE

CHAPTER 8

Indices and tables

- genindex
- search