Gluon Documentation

Release 2017.1.1

Project Gluon

Jul 19, 2017

User Documentation

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

Getting Started

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. *v2017.1.1*. 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/*.

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*
- *wget*

Building the images

To build Gluon, first check out the repository. Replace *RELEASE* with the version you'd like to checkout, e.g. *v2017.1.1*.

```
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-alpha-centauri/site-ffac.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
```

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/LEDE-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 normally not necessary, but may fix certain kinds of build failures.

```
make dirclean
```

will clean the entire tree, so the toolchain will be rebuilt as well, which will take a while.

opkg repositories

Gluon is mostly compatible with LEDE, so the normal LEDE package repositories can be used for Gluon as well.

This is not true for kernel modules; the Gluon kernel is incompatible with the kernel of the default LEDE images. Therefore, Gluon will not only generate images, but also an opkg repository containing all core packages provided by LEDE, including modules 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 public keys from two sources: the official LEDE keyring (to allow installing userspace packages) and a Gluon-specific key (which is used to sign the generated package repository).

LEDE 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.

Make variables

Gluon's build process can be controlled by various variables. They can usually be set on the command line or in `site.mk`.

Common variables

GLUON_ATH10K_MESH While Gluon does support some hardware with ath10k-based 5GHz WLAN, these WLAN adapters don't work well for meshing at the moment, so building images for these models is disabled by default. In addition, ath10k can't support IBSS and 11s meshing in the same image due to WLAN firmware restrictions.

Setting GLUON_ATH10K_MESH to `11s` or `ibss` will enable generation of images for ath10k devices and install the firmware for the corresponding WLAN mode.

GLUON_BRANCH Sets the default branch of the autoupdater. If unset, the autoupdater is disabled by default. For the `make manifest` command, GLUON_BRANCH defines the branch to generate a manifest for.

GLUON_LANGS Space-separated list of languages to include for the config mode/advanced settings. Defaults to `en`. `en` should always be included, other supported languages are `de` and `fr`.

GLUON_PRIORITY Defines the priority of an automatic update in `make manifest`. See *Autoupdater* for a detailed description of this value.

GLUON_REGION Some devices (at the moment the TP-Link Archer C7) contain a region code that restricts firmware installations. Set GLUON_REGION to `eu` or `us` to make the resulting images installable from the respective stock firmwares.

GLUON_RELEASE Firmware release number: This string is displayed in the config mode, announced via respondd/alfred and used by the autoupdater to decide if a newer version is available.

GLUON_TARGET Target architecture to build.

Special variables

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

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

GLUON_PACKAGEDIR Path where the opkg package repository will be stored. Defaults to `$(GLUON_OUTPUTDIR)/packages`.

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

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

CHAPTER 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.

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

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

Required if `next_node.ip4` is set.

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.ffac','2.ntp.services.ffac'}
```

This NTP servers must be reachable via IPv6 from the nodes. If you don't want to set an IPv6 address explicitly, but use a hostname (which is recommended), see also the *FAQ*.

opkg : optional `opkg` package manager configuration.

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

- `lede` overrides the default LEDE repository URL. The default URL would correspond to `http://downloads.lede-project.org/snapshots/packages/%A` and usually doesn't need to be changed when nodes are expected to have IPv6 internet connectivity.
- `extra` specifies a table of additional repositories (with arbitrary keys)

```
opkg = {
  lede = 'http://opkg.services.ffac/lede/snapshots/packages/%A',
  extra = {
    gluon = 'http://opkg.services.ffac/modules/gluon-%GS-%GR/%S',
  },
}
```

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

- `%n` is replaced by the LEDE version codename
- `%v` is replaced by the LEDE version number (e.g. "17.01")
- `%S` is replaced by the target board (e.g. "ar71xx/generic")
- `%A` is replaced by the target architecture (e.g. "mips_24kc")
- `%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` is 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 change the disabled state.

Additionally it is possible to configure the `supported_rates` and `basic_rate` of each radio. Both are optional, by default hostapd/driver dictate the rates. If `supported_rates` is set, `basic_rate` is required, because `basic_rate` has to be a subset of `supported_rates`. The example below disables 802.11b rates.

`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 multicast bitrate. Increasing the default value of 1000 to something like 12000 is recommended.

```
wifi24 = {
  channel = 11,
  supported_rates = {6000, 9000, 12000, 18000, 24000, 36000, 48000, 54000},
  basic_rate = {6000, 9000, 18000, 36000, 54000},
  ap = {
    ssid = 'alpha-centauri.freifunk.net',
  },
  mesh = {
    id = 'alpha-centauri-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'
}
```

The IPv4 next-node address is optional.

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,
  },
}
```

mesh_vpn Remote server setup for the mesh VPN.

The *enabled* option can be set to true to enable the VPN by default. *mtu* defines the MTU of the VPN interface.

The *fastd* section configures settings specific to the *fastd* VPN implementation.

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-web-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.

You can set syslog_level from verbose (default) to warn to reduce syslog output.

The *tunneldigger* section is used to define the *tunneldigger* broker list.

Note: It doesn't make sense to include both *fastd* and *tunneldigger* sections in the same configuration file, as only one of the packages *gluon-mesh-vpn-fastd* and *gluon-mesh-vpn-tunneldigger* should be installed with the current implementation.

```
mesh_vpn = {
  -- enabled = true,
  mtu = 1280,

  fastd = {
    methods = {'salsa2012+umac'},
    -- configurable = true,
    -- syslog_level = 'warn',
    groups = {
      backbone = {
        -- Limit number of connected peers from this group
        limit = 1,
        peers = {
          peer1 = {
            key =
↪'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX',
            -- Having multiple domains prevents SPOF in freifunk.net
            remotes = {
              'ipv4 "vpn1.alpha-centauri.freifunk.net" port 10000',
              'ipv4 "vpn1.alpha-centauri-freifunk.de" port 10000',
            },
          },
          peer2 = {
            key =
↪'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX',
            -- You can also omit the ipv4 to allow both connection via ipv4 and␣
↪ipv6
            remotes = {'"vpn2.alpha-centauri.freifunk.net" port 10000'},
          },
          peer3 = {
            key =
↪'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX',
            -- In addition to domains you can also add ip addresses, which␣
↪provides
            -- resilience in case of dns outages
            remotes = {
              '"vpn3.alpha-centauri.freifunk.net" port 10000',
              '[2001:db8::3:1]:10000',
              '192.0.2.3:10000',
            },
          },
        },
        -- Optional: nested peer groups
        -- groups = {
        --   lowend_backbone = {
        --     limit = 1,
        --     peers = ...
        --   },
        -- },
      },
      -- Optional: additional peer groups, possibly with other limits
      -- peertopeer = {
      --   limit = 10,
      --   peers = { ... },
```

```
      -- },
    },
  },

  tunneldigger = {
    brokers = {'vpn1.alpha-centauri.freifunk.net'}
  },

  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_wan = true,
```

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

```
mesh_on_lan = true,
```

poe_passthrough : optional Enable PoE passthrough by default on hardware with such a feature.

autoupdater : package Configuration for the autoupdater feature of Gluon.

The mirrors are checked in random order until the manifest could be downloaded successfully or all mirrors have been tried.

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

All configured mirrors must be reachable from the nodes via IPv6. If you don't want to set an IPv6 address explicitly, but use a hostname (which is recommended), see also the *FAQ*.

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-web-node-role` to `site.mk`.

The strings to display in the web interface are configured per language in the `i18n/en.po`, `i18n/de.po`, etc. files of the site repository using message IDs like `gluon-web-node-role:role:node` and `gluon-web-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/ffac', '/etc/config/
↪freifunk'},
       config_mode_configs = {'config_mode', 'ffac', 'freifunk'},
       fastd_configs = {'ffac_mesh_vpn', 'mesh_vpn'},
       mesh_ifname = 'freifunk',
       tc_configs = {'ffki', 'freifunk'},
       wifi_names = {'wifi_freifunk', 'wifi_freifunk5', 'wifi_mesh', 'wifi_mesh5'}
↪,
}
```

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_REGION Region code to build into images where necessary. Valid values are the empty string, `us` and `eu`.

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

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:novpn Information shown on the reboot page, if the mesh VPN was not selected.

gluon-config-mode:altitude-label Label for the `altitude` field

gluon-config-mode:altitude-help Description for the usage of the `altitude` field

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.

Site modules

The file `modules` in the site repository is completely optional and can be used to supply additional package feeds from which packages are built. The git repositories specified here are retrieved in addition to the default feeds when `make update` it called.

This file's format is very similar to the toplevel `modules` file of the Gluon tree, with the important different that the list of feeds must be assigned to the variable `GLUON_SITE_FEEDS`. Multiple feed names must be separated by spaces, for example:

```
GLUON_SITE_FEEDS='foo bar'
```

The feed names may only contain alphanumerical characters, underscores and slashes. For each of the feeds, the following variables are used to specify how to update the feed:

PACKAGES_${feed}_REPO The URL of the git repository to clone (usually `git://` or `http(s)://`)

PACKAGES_${feed}_COMMIT The commit ID of the repository to use

PACKAGES_${feed}_BRANCH Optional: The branch of the repository the given commit ID can be found in. Defaults to the default branch of the repository (usually `master`)

These variables are always all uppercase, so for an entry `foo` in GLUON_SITE_FEEDS, the corresponding configuration variables would be `PACKAGES_FOO_REPO`, `PACKAGES_FOO_COMMIT` and `PACKAGES_FOO_BRANCH`. Slashes in feed names are replaced by underscores to get valid shell variable identifiers.

Examples

site.mk

```
##        gluon site.mk makefile example

##        GLUON_SITE_PACKAGES
#                specify Gluon/LEDE packages to include here

GLUON_SITE_PACKAGES := \
        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-web-admin \
        gluon-web-autoupdater \
        gluon-web-network \
        gluon-web-wifi-config \
        gluon-mesh-batman-adv-15 \
        gluon-mesh-vpn-fastd \
        gluon-radvd \
        gluon-setup-mode \
        gluon-status-page \
        haveged \
        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')

# Variables set with ?= can be overwritten from the command line

##        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

GLUON_RELEASE ?= $(DEFAULT_GLUON_RELEASE)

# Default priority for updates.
GLUON_PRIORITY ?= 0

# Region code required for some images; supported values: us eu
```

```
GLUON_REGION ?= eu

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

site.conf

```
-- This is an example site configuration for Gluon v2017.1.1
--
-- 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 Alpha Centauri',

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

  -- Prefixes used within the mesh.
  -- prefix6 is required, prefix4 can be omitted if next_node.ip4
  -- is not set.
  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,

    -- List of supported wifi rates (optional)
    -- Example removes 802.11b compatibility for better performance
    supported_rates = {6000, 9000, 12000, 18000, 24000, 36000, 48000, 54000},

    -- List of basic wifi rates (optional, required if supported_rates is set)
    -- Example removes 802.11b compatibility for better performance
    basic_rate = {6000, 9000, 18000, 36000, 54000},

    -- ESSID used for client network.
    ap = {
```

```
    ssid = 'alpha-centauri.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 = 'alpha-centauri.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,
  -- },
-- },

mesh_vpn = {
  -- enabled = true,
  mtu = 1280,

  fastd = {
    -- Refer to http://fastd.readthedocs.org/en/latest/ to better understand
    -- what these options do.

    -- List of crypto-methods to use.
    methods = {'salsa2012+umac'},
```

```
      -- configurable = true,
      -- syslog_level = 'warn',

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

          -- List of peers.
          peers = {
            peer1 = {
              key = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
↪',

              -- This is a list, so you might add multiple entries.
              remotes = {'ipv4 "xxx.somehost.invalid" port xxxxxx'},
            },
            peer2 = {
              key = 'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
↪',
              -- 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,
  -- },

  -- config_mode = {
    -- Show/hide the altitude field
    -- geo_location = {
      -- show_altitude = false,
    -- },
    -- define if the contact field is obligatory (optional)
    -- owner = {
      -- obligatory = 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 Alpha Centauri 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 servers. "
"To register, send the key together with your node's name (<em><%=pcdata(hostname)%></
↪em>) to "
"<a href=\"mailto:keys@alpha-centauri.freifunk.net?subject=<%= urlencode(
↪'Registration: ' .. hostname) %>&amp;"
"body=<%= urlencode('# ' .. hostname .. '\n' .. pubkey) %>\">keys@alpha-centauri.
↪freifunk.net</a>."
"</p>"
"<div class=\"the-key\">"
" # <%= pcdata(hostname) %>"
" <br />"
"<%= pubkey %>"
"</div>"

msgid "gluon-config-mode:novpn"
msgstr ""
"<p>You have selected <strong>not</strong> to use the mesh VPN. "
"Your node will only be able to connect to the Freifunk network if other nodes in␣
↪reach "
"already have a connection.</p>"

msgid "gluon-config-mode:reboot"
msgstr ""
"<p>Your node <em><%= pcdata(hostname) %></em> is currently rebooting and will "
"try to connect to other nearby Freifunk nodes after that. For more "
"information about the Freifunk community on Alpha Centauri, have a look at "
"<a href=\"https://alpha-centauri.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>"

msgid "gluon-config-mode:altitude-label"
msgstr "Altitude"

msgid "gluon-config-mode:altitude-help"
```

```
msgstr ""
"Specifying the altitude is optional and should only be done if a proper "
"value is known."
```

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 Alpha Centauri "
"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 Freifunk-Projektes auf Alpha Centauri eingetragen wurde, "
"kann sich dein Knoten mit dem Mesh-VPN dort verbinden. Bitte "
"schicke dazu diesen Schlüssel und den Namen deines Knotens "
"(<em><%=pcdata(hostname)%></em>) an "
"<a href=\"mailto:keys@alpha-centauri.freifunk.net?subject=<%= urlencode('Anmeldung:
↪' .. hostname) %>&amp;"
"body=<%= urlencode('# ' .. hostname .. '\n' .. pubkey) %>\">keys@alpha-centauri.
↪freifunk.net</a>."
"</p>"
"<div class=\"the-key\">"
" # <%= pcdata(hostname) %>"
" <br />"
"<%= pubkey %>"
"</div>"

msgid "gluon-config-mode:novpn"
msgstr ""
"<p>Du hast ausgewählt, <strong>kein Mesh-VPN</strong> "
"zu nutzen. Dein Knoten kann also nur dann eine Verbindung zum Freifunk-Netz "
"aufbauen, wenn andere Freifunk-Router in WLAN-Reichweite sind."
"</p>"

msgid "gluon-config-mode:reboot"
msgstr ""
"<p>Dein Knoten <em><%= pcdata(hostname) %></em> startet gerade neu und wird "
"anschließend versuchen, sich mit anderen Freifunkknoten in seiner Nähe zu "
"verbinden. Weitere Informationen zur "
"Alpha Centauri Freifunk-Community findest du auf "
"<a href=\"https://alpha-centauri.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>"

msgid "gluon-config-mode:altitude-label"
msgstr "Höhe"

msgid "gluon-config-mode:altitude-help"
msgstr ""
"Die Höhenangabe ist optional und sollte nur gesetzt werden, wenn ein "
"exakter Wert bekannt ist."
```

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-ffac (Regio Aachen)
- site-ffbs (Braunschweig)
- site-ffhb (Bremen)
- site-ffda (Darmstadt)
- site-ffeh (Ehingen)
- site-fffl (Flensburg)
- site-ffgoe (Göttingen)

- site-ffgt-rhw (Guetersloh)
- site-ffhh (Hamburg)
- site-ffho (Hochstift)
- site-ffhgw (Greifswald)
- site-ffka (Karlsruhe)
- site-ffki (Kiel)
- site-fflz (Lausitz)
- site-ffl (Leipzig)
- site-ffhl (Lübeck)
- site-fflg (Lüneburg)
- site-ffmd (Magdeburg)
- site-ffmwu (Mainz, Wiesbaden & Umgebung)
- site-ffmyk (Mayen-Koblenz)
- site-ffmo (Moers)
- site-ffmg (Mönchengladbach)
- site-ffm (München)
- site-ffhmue (Münden)
- site-ffms (Münsterland)
- site-neuss (Neuss)
- site-ffniers (Niersufer)
- site-ffnw (Nordwest)
- site-ffrgb (Regensburg)
- site-ffrn (Rhein-Neckar)
- site-ffruhr (Ruhrgebiet, Multi-Communities)
- site-ffs (Stuttgart)
- site-fftr (Trier)

CHAPTER 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.

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.

x86-geode x86 image for Geode CPUs.

x86-64 64bit version of *x86-generic*.

CHAPTER 4

Frequently Asked Questions

DNS does not work on the nodes

Gluon nodes will ignore the DNS server on the WAN port for everything except the mesh VPN, which can lead to confusion.

All normal services on the nodes exclusively use the DNS server on the mesh interface. This DNS server must be announced in router advertisements (using *radvd* or a similar software) from one or more central servers in meshes based on *batman-adv*. If your mesh does not have global IPv6 connectivity, you can setup your *radvd* not to announce a default route by setting the *default lifetime* to 0; in this case, the *radvd* is only used to announce the DNS server.

CHAPTER 5

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:

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.

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.

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.

CHAPTER 6

Autoupdater

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

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, the `PRIORITY` value may be defined by setting `GLUON_PRIORITY` on the command line or in `site.mk`.

`GLUON_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 declared in the manifest file by the variable DATE and then slowly rise up to 1 when `GLUON_PRIORITY` days have passed. The autoupdater checks for updates hourly (at a random minute of the hour), but usually only updates during its run between 4am and 5am, except when the whole `GLUON_PRIORITY` days and another 24 hours have passed.

`GLUON_PRIORITY` may be an integer or a decimal fraction.

Automated nightly builds

A fully automated nightly build could use the following commands:

```
git pull
(cd site && git pull)
make update
make clean
NUM_CORES_PLUS_ONE=$(expr $(nproc) + 1)
make -j$NUM_CORES_PLUS_ONE 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
```

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.

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
```

```
# If fallback is true the updater will perform an update only if the timespan
# PRIORITY days (as defined in the manifest) and another 24h have passed
autoupdater --fallback
```

CHAPTER 7

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.

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 it 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.

During upgrades the wifi channel of the 2.4GHz and 5GHz radio will be restored to the channel configured in the site.conf. If you need to preserve a user defined wifi channel during upgrades you can configure this via the uci section `gluon-core.wireless`:

```
uci set gluon-core.@wireless[0].preserve_channels='1'
```

Keep in mind that nodes running wifi interfaces on custom channels can't mesh with default nodes anymore!

CHAPTER 8

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-web-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
```

CHAPTER 9

Wired mesh (Mesh-on-WAN/LAN)

In addition to meshing over WLAN and VPN, it is also possible to configure 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).

Configuration

Both Mesh-on-WAN and Mesh-on-LAN can be configured on the "Network" page of the *Advanced settings* (if the package `gluon-web-network` 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 network
```

It may be disabled by running:

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

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 network
```

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 network
```

Please note that this configuration has changed in Gluon v2016.1. Using the old commands on v2016.1 and later will break the corresponding options in the *Advanced settings*.

CHAPTER 10

DNS caching

User experience may be greatly improved when dns is accelerated. Also, it seems like a good idea to keep the number of packages being exchanged between node and gateway as small as possible. In order to do this, a DNS cache may be used on a node. The dnsmasq instance listening on port 53 on the node will be reconfigured to answer requests, use a list of upstream servers and a specific cache size if the options listed below are added to site.conf. Upstream servers are the DNS servers which are normally used by the nodes to resolve hostnames (e.g. gateways/supernodes).

There are the following settings: servers cacheentries

If both options are set the node will cache as much DNS records as set with 'cacheentries' in RAM. The 'servers' list will be used to resolve the received DNS queries if the request cannot be answered from cache. If these settings do not exist, the cache is not intialized and RAM usage will not increase.

When next_node.name is set, an A record and an AAAA record for the next-node IP address are placed in the dnsmasq configuration. This means that the content of next_node.name may be resolved even without upstream connectivity.

```
dns = {
  cacheentries = 5000,
  servers = { '2001:db8::1', },
},

next_node = {
  name = 'nextnode',
  ip6 = '2001:db8:8::1',
  ip4 = '198.51.100.1',
}
```

The cache will be initialized during startup. Each cache entry will occupy about 90 bytes of RAM.

CHAPTER 11

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.

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.

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.

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.

CHAPTER 12

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.

CHAPTER 13

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',
    'test',
    'backbone',
    'service',
  },
},
```

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

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-web-node-role` to your `site.mk`.

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 14

Development Basics

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

Bug Tracker

The main repo does have issues enabled.

IRC

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

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*.

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!

Development Guidelines

lua should be used instead of sh whenever sensible. The following criteria should be considered:

- Is the script doing more than just executing external commands? if so, use lua
- Is the script parsing/editing json-data? If so, use lua for speed
- When using sh, use jsonfilter instead of json_* functions for speed

Code formatting may sound like a topic for the pedantic, however it helps if the code in the project is formatted in the same way. The following rules apply:

- use tabs instead of spaces
- trailing whitespaces must be eliminated

CHAPTER 15

Adding support for new hardware

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

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).

Adding profiles

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

Profiles are defined in `targets/*` in a shell-based DSL (so common shell commands syntax like `if` can be used.

The `device` command is used to define an image build for a device. It takes two or three parameters.

The first parameter defines the Gluon profile name, which is used to refer to the device and is part of the generated image name. The profile name must be same as the output of the following command (on the target device), so the autoupdater can work:

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

The second parameter defines the name of the image files generated by LEDE. Usually, it is also the LEDE profile name; for devices that still use the old image build code, a third parameter with the LEDE profile name can be passed. The profile names can be found in the image Makefiles in `lede/target/linux/<target>/image/Makefile`.

Examples:

```
device tp-link-tl-wr1043n-nd-v1 tl-wr1043nd-v1
device alfa-network-hornet-ub hornet-ub HORNETUB
```

Suffixes and extensions

By default, image files are expected to have the extension `.bin`. In addition, the images generated by LEDE have a suffix before the extension that defaults to `-squashfs-factory` and `-squashfs-sysupgrade`.

This can be changed using the `factory` and `sysupgrade` commands, either at the top of the file to set the defaults for all images, or for a single image. There are three forms with 0 to 2 arguments (all work with `sysupgrade` as well):

```
factory SUFFIX .EXT
factory .EXT
factory
```

When only an extension is given, the default suffix is retained. When no arguments are given, this signals that no factory (or sysupgrade) image exists.

Aliases

Sometimes multiple models use the same LEDE images. In this case, the `alias` command can be used to create symlinks and additional entries in the autoupdater manifest for the alternative models.

Standalone images

On targets without *per-device rootfs* support in LEDE, the commands described above can't be used. Instead, `factory_image` and `sysupgrade_image` are used:

```
factory_image PROFILE IMAGE .EXT
sysupgrade_image PROFILE IMAGE .EXT
```

Again, the profile name must match the value printed by the aforementioned Lua command. The image name must match the part between the target name and the extension as generated by LEDE and is to be omitted when no such part exists.

Packages

The `packages` command takes an arbitrary number of arguments. Each argument defines an additional package to include in the images in addition to the default package sets defined by LEDE. When a package name is prefixed by a minus sign, the packages are excluded instead.

The `packages` command may be used at the top of a target definition to modify the default package list for all images, or just for a single device (when the target supports *per-default rootfs*).

Configuration

The `config` command allows to add arbitary target-specific LEDE configuration to be emitted to `.config`.

Notes

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/010-primary-mac` 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 a LEDE bug was the cause).

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:

Package adjustments

One package that may need adjustments for new targets is `libplatforminfo` (to be found in packages/gluon/libs/libplatforminfo). If the new platform works fine with the definitions found in `default.c`, nothing needs to be done. Otherwise, create a definition for the added target or subtarget, either by symlinking one of the files in the `templates` directory, or adding a new source file.

On many targets, Gluon's network setup scripts (mainly in the package `gluon-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).

Build system support

A definition for the new target must be created under `targets`, and it must be added to `targets/targets.mk`. The `GluonTarget` macro takes one to three arguments: the target name, the Gluon subtarget name (if the target has subtargets), and the LEDE subtarget name (if it differs from the Gluon subtarget). The third argument can be used to define multiple Gluon targets with different configuration for the same LEDE target, like it is done for the `ar71xx-tiny` target.

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

CHAPTER 16

Upgrade scripts

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. Note that the script files need to be executable.

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.

Best practices

- Most upgrade scripts are written in Lua. This allows using lots of helper functions provided by 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 achieve 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 is `nil` during the initial installation and contains the previously install Gluon version otherwise.

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

CHAPTER 17

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.

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.

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.

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.

CHAPTER 18

MAC addresses

Many devices don't have enough unique MAC addresses assigned by the vendor (in batman-adv, each mesh interface needs an own MAC address that must be unique mesh-wide).

Gluon tries to solve this issue by using a hash of the primary MAC address as a 45 bit MAC address prefix. The resulting 8 addresses are used as follows:

- 0: client0; WAN
- 1: mesh0
- 2: ibss0
- 3: wan_radio0 (private WLAN); batman-adv primary address
- 4: client1; LAN
- 5: mesh1
- 6: ibss1
- 7: wan_radio1 (private WLAN); mesh VPN

CHAPTER 19

Controllers

Controllers live in `/lib/gluon/web/controller`. They define which pages ("routes") exist under the `/cgi-bin/gluon` path, and what code is run when these pages are requested.

Controller scripts mostly consist of calls of the *entry* function, which each define one route:

```
entry({"admin"}, alias("admin", "info"), _("Advanced settings"), 10)
entry({"admin", "info"}, template("admin/info"), _("Information"), 1)
```

The entry function expects 4 arguments:

- *path*: Components of the path to define a route for.

  The above example defines routes for the paths `admin` and `admin/info`.

- *target*: Dispatcher for the route. See the following section for details.
- *title*: Page title (also used in navigation). The underscore function is used
- *order*: Sort index in navigation (defaults to 100)

Navigation indexes are automatically generated for each path level. Pages can be hidden from the navigation by setting the *hidden* property of the node object returned by *entry*:

```
entry({"hidden"}, alias("foo"), _("I'm hidden!")).hidden = true
```

Dispatchers

- *alias* (*path*, ...): Redirects to a different page. The path components are passed as individual arguments.
- *call* (*func*, ...): Runs a Lua function for custom request handling. The given function is called with the HTTP object and the template renderer as first two arguments, followed by all additional arguments passed to *call*.
- *template* (*view*): Renders the given view. See *Views*.
- *model* (*name*): Displays and evaluates a form as defined by the given model. See the *Models* page for an explanation of gluon-web models.

The HTTP object

The HTTP object provides information about the HTTP requests and allows to add data to the reply. Using it directly is rarely necessary when gluon-web models and views are used.

Useful functions:

- *getenv* (*key*): Returns a value from the CGI environment passed by the webserver.
- *formvalue* (*key*): Returns a value passed in a query string or in the content of a POST request. If multiple values with the same name have been passed, only the first is returned.
- *formvaluetable* (*key*): Similar to *formvalue*, but returns a table of all values for the given key.
- *status* (*code*, *message*): Writes the HTTP status to the reply. Has no effect if a status has already been sent or non-header data has been written.
- *header* (*key*, *value*): Adds an HTTP header to the reply to be sent to to the client. Has no effect when non-header data has already been written.
- *prepare_content* (*mime*): Sets the *Content-Type* header to the given MIME type, potentially setting additional headers or modifying the MIME type to accommodate browser quirks
- *write* (*data*, ...): Sends the given data to the client. If headers have not been sent, it will be done before the data is written.

HTTP functions are called in method syntax, for example:

```
http:write('Output!')
```

The template renderer

The template renderer allows to render templates (views). The most useful functions are:

- *render* (*view*, *scope*): Renders the given view, optionally passing a table with additional variables to make available in the template.
- *render_string* (*str*, *scope*): Same as *render*, but the template is passed directly instead of being loaded from the view directory.

The renderer functions are called in property syntax, for example:

```
renderer.render('layout')
```

Differences from LuCI

- Controllers must not use the *module* function to define a Lua module (*gluon-web* will set up a proper environment for each controller itself)
- Entries are defined at top level, not inside an *index* function
- The *alias* dispatcher triggers an HTTP redirect instead of directly running the dispatcher of the aliased route.
- The *call* dispatcher is passed a function instead of a string with a function name.
- The *cbi* dispatcher of LuCI has been renamed to *model*.

- The HTTP POST handler support the multipart/form-data encoding only, so `enctype="multipart/form-data"` must be included in all <*form*> HTML elements.
- Other dispatchers like *form* are not provided.

CHAPTER 20

Models

Models are defined in `/lib/gluon/web/model`. Each model defines one or more forms to display on a page, and how the submitted form data is handled.

Let's start with an example:

```
local f = Form(translate('Hostname'))

local s = f:section(Section)

local o = s:option(Value, 'hostname', translate('Hostname'))
o.default = uci:get_first('system', 'system', 'hostname')
function o:write(data)
  uci:set('system', uci:get_first('system', 'system'), 'hostname', data)
  uci:commit('system')
end

return f
```

The toplevel element of a model is always a *Form*, but it is also possible for a model to return multiple forms, which are displayed one below the other.

A *Form* has one or more *Sections*, and each *Section* has different types of options.

All of these elements have an *id*, which is used to identify them in the HTML form and handlers. If no ID is given, numerical IDs will be assigned automatically, but using explicitly named elements is often advisable (and it is required if a form does not always include the same elements, i.e., some forms, sections or options are added conditionally). IDs are hierarchical, so in the above example, the *Value* would get the ID `1.1.hostname` (value *hostname* in first section of first form).

Classes and methods

- *Form* (*title*, *description*, *id*)

  - *Form:section* (*type*, *title*, *description*, *id*)

    Creates a new section of the given type (usually *Section*).

  - *Form:write* ()

    Is called after the form has beed submitted (but only if the data is valid). It is called last (after all options' *write* methods) and is usually used to commit changed UCI packages.

    The default implementation of *write* doesn't to anything, but it can be overridden.

- *Section* (usually instanciated through *Form:section*)
  - *Section:option* (*type*, *id*, *title*, *description*)

    Creates a new option of the given type. Option types:

    * *Value*: simple text entry
    * *TextValue*: multiline text field
    * *ListValue*: radio buttons or dropdown selection
    * *DynamicList*: variable number of text entry fields
    * *Flag*: checkbox

Most option types share the same properties and methods:

- *default*: default value
- *optional*: value may be empty
- *datatype*: one of the types described in *Data types*

  By default (when *datatype* is *nil*), all values are accepted.

- *state*: has one of the values *FORM_NODATA*, *FORM_VALID* and *FORM_INVALID* when read in a form handler

  An option that has not been submitted because of its dependencies will have the state *FORM_NODATA*, *FORM_INVALID* if the submitted value is not valid according to the set *datatype*, and *FORM_VALID* otherwise.

- *data*: can be read in form handlers to get the submitted value
- *depends* (*self*, *option*, *value*): adds a dependency on another option

  The option will only be shown when the passed option has the given value. This is mainly useful when the other value is a *Flag* or *ListValue*.

- *depends* (*self*, *deps*): adds a dependency on multiple other options

  deps must be a table with options as keys and values as values. The option will only be shown when all passed options have the corresponding values.

  Multiple alternative dependencies can be added by calling *depends* repeatedly.

- *value* (*self*, *value*, *text*): adds a choice to a *ListValue*
- *write* (*self*, *data*): is called with the submitted value when all form data is valid.

  Does not do anything by default, but can be overridden.

The *default* value, the *value* argument to *depends* and the output *data* always have the same type, which is usually a string (or *nil* for optional values). Exceptions are:

- *Flag* uses boolean values
- *DynamicList* uses a table of strings

Despite its name, the *datatype* setting does not affect the returned value type, but only defines a validator the check the submitted value with.

For a more complete example that actually makes use of most of these features, have a look at the model of the *gluon-web-network* package.

Data types

- *integer*: an integral number
- *uinteger*: an integral number greater than or equal to zero
- *float*: a number
- *ufloat*: a number greater than or equal to zero
- *ipaddr*: an IPv4 or IPv6 address
- *ip4addr*: an IPv4 address
- *ip6addr*: an IPv6 address
- *wpakey*: a string usable as a WPA key (either between 8 and 63 characters, or 64 hex digits)
- *range* (*min*, *max*): a number in the given range (inclusive)
- *min* (*min*): a number greater than or equal to the given minimum
- *max* (*max*): a number less than or equal to the given maximum
- *irange* (*min*, *max*): an integral number in the given range (inclusive)
- *imin* (*min*): an integral number greater than or equal to the given minimum
- *imax* (*max*): an integral number less than or equal to the given maximum
- *minlength* (*min*): a string with the given minimum length
- *maxlength* (*max*): a string with the given maximum length

Differences from LuCI

- LuCI's *SimpleForm* and *SimpleSection* are called *Form* and *Section*, respectively
- Is it not possible to add options to a *Form* directly, a *Section* must always be created explicitly
- Many of LuCI's CBI classes have been removed, most importantly the *Map*
- The *rmempty* option attribute does not exist, use *optional* instead
- Only the described data types are supported
- Form handlers work completely differently (in particular, a *Form*'s *handle* method should usually not be overridden in *gluon-web*)

CHAPTER 21

Views

The template parser reads views from `/lib/gluon/web/view`. Writing own view should be avoided in favour of using *Models* with their predefined views.

Views are partial HTML pages, with additional template tags that allow to embed Lua code and translation strings. The following tags are defined:

- `<% ... %>` evaluates the enclosed Lua expression.
- `<%= ... %>` evaluates the enclosed Lua expression and prints its value.
- `<%+ ... %>` includes another template.
- `<%: ... %>` translates the enclosed string using the loaded i18n catalog.
- `<%_ ... %>` translates the enclosed string *without escaping HTML entities* in the translation. This only makes sense when the i18n catalog contains HTML code.
- `<%# ... %>` is a comment.

All of these also come in the whitespace-stripping variants `<%-` and `-%>` that remove all whitespace before or after the tag.

Complex combinations of HTML and Lua code are possible, for example:

```
<div>
  <% if foo then %>
    Content
  <% end %>
</div>
```

Variables and functions

Many call sites define additional variables (for example, model templates can access the model as *self* and a unique element ID as *id*), but the following variables and functions should always be available for the embedded Lua code:

- *renderer*: *The template renderer*

- *http*: *The HTTP object*
- *request*: Table containing the path components of the current page
- *url* (*path*): returns the URL for the given path, which is passed as a table of path components.
- *attr* (*key*, *value*): Returns a string of the form `key="value"` (with a leading space character before the key).

  value is converted to a string (tables are serialized as JSON) and HTML entities are escaped. Returns an empty string when *value* is *nil* or *false*.
- *include* (*template*): Includes another template.
- *node* (*path*, ...): Returns the controller node for the given page (passed as one argument per path component).

  Use `node(unpack(request))` to get the node for the current page.
- *pcdata* (*str*): Escapes HTML entities in the passed string.
- *urlencode* (*str*): Escapes the passed string for use in an URL.
- *translate*, *_translate* and *translatef*: see *Internationalization support*

CHAPTER 22

Internationalization support

General guidelines

- All config mode packages must be fully translatable, with complete English and German texts.
- All new expert mode packages must be fully translatable. English texts are required.
- German translations are recommended. Other supported languages, especially French, are nice-to-have, but not required. If you don't know a language well, rather leave the translation blank, so it is obvious that there is no proper translation yet.
- 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.

i18n support in Gluon

Internationalization support is available in all components (models, view and controllers) of *gluon-web*-based packages. Strings are translated using the *translate*, *_translate* and *translatef* functions (*translate* for static strings, *translatef* for printf-like formatted string; *_translate* works the same as *translate*, but will return *nil* instead of the original string when no translation is available) . In views, the special tags `<%: ... %>` can be used to translate the contained string.

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

```
local share_location = s:option(Flag, "location", translate("Show node on the map"))
```

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 in the `contrib` directory:

```
cd package/gluon-web-mesh-vpn-fastd
mkdir i18n
cd i18n
../../../contrib/i18n-scan.pl ../files ../luasrc > gluon-web-mesh-vpn-fastd.pot
```

The same command can be run again to update the template.

In addition, some additions to the Makefile must be made. Instead of LEDE's default *package.mk*, the Gluon version (`../gluon.mk` for core packages) must be used. The i18n files must be installed and PKG_CONFIG_DEPENDS must be added:

```
...
include ../gluon.mk

PKG_CONFIG_DEPENDS += $(GLUON_I18N_CONFIG)
...
define Build/Compile
  $(call GluonBuildI18N,gluon-web-mesh-vpn-fastd,i18n)
endef

define Package/gluon-web-mesh-vpn-fastd/install
  ...
  $(call GluonInstallI18N,gluon-web-mesh-vpn-fastd,$(1))
endef
...
```

Adding translations

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

```
cd package/gluon-web-mesh-vpn-fastd/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-web-mesh-vpn-fastd.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 translation file should be translated in the *.po* file (or removed from it, so the original English texts are displayed instead).

Adding support for new languages

A list of all languages supported by *gluon-web* can be found in `package/gluon.mk`. New languages just need to be added to *GLUON_SUPPORTED_LANGS*, after a human-readable language name has been defined in the same file.

CHAPTER 23

Config Mode

The *Config Mode* consists of several modules that provide a range of different condiguration options:

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.

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. In the standard package set, the order is, for wizard modules:

- 0050-autoupdater-info
- 0100-hostname
- 0300-mesh-vpn
- 0400-geo-location
- 0500-contact-info

The reboot module order is:

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

All modules are run in the gluon-web model context and have access to the same variables as "full" gluon-web modules.

Wizards

Wizard modules must return a function that is provided with the wizard form and an UCI cursor. The function can create configuration sections in the form:

```
return function(form, uci)
  local s = form:section(Section)
  local o = s:option(Value, "hostname", "Hostname")
  o.default = uci:get_first("system", "system", "hostname")
  o.datatype = "hostname"

  function o:write(data)
    uci:set("system", uci:get_first("system", "system"), "hostname", data)
  end

  return {'system'}
end
```

The function may return a table of UCI packages to commit after the individual fields' *write* methods have been executed. This is done to avoid committing the packages repeatedly when multiple wizard modules modify the same package.

Reboot page

Reboot modules are simply executed when the reboot page is rendered:

```
renderer.render_string("Hello World!")
```

CHAPTER 24

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*.

site.conf

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

CHAPTER 25

gluon-config-mode-contact-info

This package allows the user to provide contact information within config mode to be distributed in the mesh. You can define whether the owner contact field is obligatory or not in your site.conf.

site.conf

config_mode.owner.obligatory : optional (defaults to false) If `obligatory` is set to `true`, the contact info field must be supplied and may not be left empty.

Example:

```
config_mode = {
  owner = {
    obligatory = true
  }
}
```

CHAPTER 26

gluon-config-mode-geo-location

This package enables the user to set latitude, longitude and altitude of their node within config mode. As the usage of the altitude is not well defined the corresponding field can be disabled.

site.conf

config_mode.geo_location.show_altitude [optional]

- `true` enables the altitude field
- `false` disables the altitude field if altitude has not yet been set
- defaults to `true`

CHAPTER 27

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)

CHAPTER 28

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 29

gluon-ebtables-segment-mld

These filters drop IGMP/MLD packets before they enter the mesh and filter any IGMP/MLD packets coming from the mesh.

IGMP/MLD have the concept of a local, elected Querier. For more decentralization and increased robustness, the idea of this package is to split the IGMP/MLD domain a querier is responsible for, allowing to have a querier per node. The split IGMP/MLD domain will also reduce overhead for this packet type, increasing scalability.

Beware of the consequences of using this package though: You might need to explicitly, manually mark ports on snooping switches leading towards your mesh node as multicast router ports for now (Multicast Router Discovery, MRD, not implemented yet).

CHAPTER 30

gluon-ebtables-source-filter

The *gluon-ebtables-source-filter* package adds an additional layer-2 filter ruleset to prevent unreasonable traffic entering the network via the nodes. Unreasonable means traffic entering the mesh via a node which source IP does not belong to the configured IP space.

One may first check if there is a certain proportion of unreasonable traffic, before adding this package to the firmware image. Additional one should not use this package if some kind of gateway or upstream network is provided by a device connected to the client port.

site.conf

prefix4 [optional]

- IPv4 subnet

prefix6 :

- IPv6 subnet

extra_prefixes6 [optional]

- list of additional IPv6 subnets

Example:

```
prefix4 = '198.51.100.0/21',
prefix6 = '2001:db8:8::/64',
extra_prefixes6 = { '2001:db8:9::/64', '2001:db8:100::/60' },
```

CHAPTER 31

Gluon 2017.1.1

Bugfixes

- The autoupdater manifest has been extended to allow automatic upgrades from old *x86-kvm* and *x86-xen_domu* systems to the new *x86-generic* image (869ceb4)
- Make flash writable again on Ubiquiti PicoStations with certain bootloader versions (and possibly other devices) (9a787c9)

  Units affected by this issue running Gluon v2017.1 can't leave config mode and no regular sysupgrades are possible. TFTP recovery is necessary to make them work again.
- Add workaround to prevent sporadic segfaults of busybox (ash) when running shell scripts on ar71xx (#1157)
- Disable batman-adv multicast optimizations to work around issue causing large amounts of management traffic (819758f)

  Multicast optimizations will be enabled again when a proper fix is available.

Known issues

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

  Reducing the TX power in the Advanced Settings 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.

CHAPTER 32

Gluon 2017.1

General changes

Gluon 2017.1 is the first release of Gluon based on the LEDE 17.01 branch. The kernel has been updated from 3.18.x to 4.4.x.

We've used the opportunity to greatly simplify the Gluon build system, removing many hacks that were required to make the build work with older OpenWrt releases.

The *output/modules* directory is now called *output/packages* and provides a replacement for the whole repository with target-specific packages of LEDE (in contrast to packages that are common for all targets of the same architecture). Another change to the build system makes it necessary that the same *GLUON_RELEASE* value that is used to build the images is also set for `make manifest`.

GCC 4.8 or newer is now required to build Gluon.

Note: There is an issue in all Gluon versions before 2016.2.6 that will lead to x86 systems losing their configuration when upgrading to Gluon 2017.1! Older Gluon versions should be upgraded to 2016.2.6 first before switching to 2017.1.

Another potential issue mostly affects virtual machines: Gluon 2017.1 images are bigger than 2016.2.x images on x86. If your virtual harddisk is based on a 2016.2.x image, it must be resized to 273MB or bigger before upgrading to Gluon 2017.1. Using qemu, the command

```
qemu-img resize $IMAGE 273MB
```

can be used to do this.

Added hardware support

ar71xx-generic

- TP-Link

  - RE450
  - WBS210 v1.20
  - WBS510 v1.20
- Ubiquiti
  - AirGateway LR
  - AirGateway PRO
  - Rocket M2/M5 Ti
  - UniFi AP LR

ar71xx-tiny

The new *ar71xx-tiny* target has split out of *ar71xx-generic*; all *ar71xx-generic* devices with only 4MB of flash have been moved to this target.

In contrast to *ar71xx-generic*, *ar71xx-tiny* **does not support opkg anymore** to save some space.

- TP-Link
  - TL-WA730RE v1
  - TL-WA7210N v2

x86-generic

The *x86-kvm* and *x86-xen_domu* targets have been removed; the *x86-generic* images now support these usecases as well, so no separate targets are needed anymore.

x86-geode

The new *x86-geode* target for hardware based on Geode CPUs has been added.

New features

- Localization support has been added to the status page. In addition to German, there are English and Russian translations now (#1044)
- Add support for making nodes a DNS cache for clients (#1000)

  See also: *DNS caching*
- Add L2TP via tunneldigger as an alternative VPN system (#978)

  L2TP will usually give better performance than fastd as it runs in kernel space, but it does not provide encryption. Also, tunneling over IPv6 is currently unsupported by tunneldigger.

  It is not possible to include both fastd and tunneldigger in the same firmware.
- Add source filter package (#1015)

  The new package *gluon-ebtables-source-filter* can be used to prevent traffic using unexpected IP addresses or packet types from entering the mesh.

See also: *gluon-ebtables-source-filter*

Bugfixes

- Disabling batman-adv on an interface (for example when an Ethernet link is lost or before sysupgrades) could lead to a kernel crash in certain configurations (#680)
- A race condition in the network setup scripts could lead to incomplete setup during boot or when interfaces were added or removed from batman-adv after Ethernet link changes (#905)

  The fix also solved the long-standing issue of Ethernet-only nodes (i.e. no WLAN or VPN mesh) not booting up correctly without an Ethernet mesh link.
- Some fixes in the WLAN stack of LEDE have improved the stability of the ath9k driver (#605)

Site changes

site.mk

- The *gluon-legacy* package does not exist anymore
- All *gluon-luci-* packages have been renamed to *gluon-web-*; *gluon-luci-portconfig* is now called *gluon-web-network*
- The *gluon-next-node* package has been merged into the Gluon core and must not be specified in *site.mk* anymore

site.conf

- The *fastd_mesh_vpn* configuration section has been restructured to allow sharing more options with tunneldigger. Instead of

```
fastd_mesh_vpn = {
  mtu = 1280,
  configurable = true,
  methods = {'salsa2012+umac'},
  groups = { ... },
  bandwidth_limit = { ... },
}
```

  the configuration must look like this now:

```
mesh_vpn = {
  mtu = 1280,
  fastd = {
    configurable = true,
    methods = {'salsa2012+umac'},
    groups = { ... },
  }
  bandwidth_limit = { ... },
}
```

- The *opkg.openwrt* option has been renamed to *opkg.lede*

i18n

- The *escape* function has been removed as it was duplicating the existing *pcdata* function. All uses of *escape* in i18n templates must be changed to use *pcdata* instead.
- The *gluon-config-mode:altitude-label* and *gluon-config-mode:altitude-help* translation IDs have been added to allow adjusting the texts for different kinds of altitudes that might be expected.
- The optional *gluon-config-mode:novpn* label has been added, which will be shown in place of *gluon-config-mode:pubkey* when mesh VPN is disabled.

Internals

- The LuCI base libraries have been replaced by a stripped-down version called "gluon-web" (#1007)

  Custom packages will need to be adjusted; in particular, all uses of *luci.model.uci* need to be replaced with *simple-uci*. The Gluon documentation explains the most important changes required to migrate from LuCI to gluon-web.
- respondd now listens on `ff05::2:1001` in addition to `ff02::2:1001` for mesh-wide operation (#984)

  Eventually, `ff02::2:1001` will be available for exchanging information between neighbouring nodes only; map servers should be moved to `ff05::2:1001`.
- batman-adv has been updated to version 2017.1
- Directly running make commands in the *lede* directory is supported now. Consequently, build targets like `target/linux/clean` and `package/NAME/compile` can't be used in the Gluon repository root anymore.

  The command `make config` will set up the LEDE *.config* in the way a normal Gluon build would, so it's possible to build individual packages for testing and development afterwards.
- Target definitions have been migrated from a Make-based format to a simpler shell-based DSL
- Gluon does not pass any custom variables into the LEDE build anymore, so things like *GLUONDIR*, *GLUON_VERSION*, or *GLUON_SITEDIR* aren't available to package Makefiles in Gluon 2017.1.

  Instead of `$(GLUONDIR)/package.mk`, `$(TOPDIR)/../package/gluon.mk` must be included in custom packages now.

Known issues

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

  Reducing the TX power in the Advanced Settings 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.

CHAPTER 33

Gluon 2016.2.6

Added hardware support

ar71xx-generic

- TP-Link TL-WR841N/ND v12

Bugfixes

- Fix CVE-2016-10229 (#1097)

  Fortunately, the standard Gluon setup is not vulnerable, as the issue only affects applications that use MSG_PEEK on UDP sockets. dnsmasq does use MSG_PEEK, but only in the DHCP component, which is not enabled during normal node operation.

- Fix roaming issue affecting communication between clients (#1121)

  This issue affects all previous releases of Gluon v2016.2.x.

- Fix build against OpenSSL 1.1 (b6a22ce)
- Fix build with long path names (#1120)
- Use new staged sysupgrade procedure (d4a69c0)

  The new sysupgrade fixes an issue affecting x86, causing nodes to lose their configuration on upgrade when the size of the kernel partition grows. This is the case when upgrading from Gluon v2016.2.x to newer (LEDE-based) Gluon versions. **This means that a Gluon node running an older version must be upgraded to Gluon v2016.2.6 first before switching to a LEDE-based version!**

  One downside of the staged sysupgrade is that all processes, including the SSH server, will be terminated at the start of the sysupgrade to allow unmounting the root filesystem. This makes it impossible to get any feedback from the upgrade process without a serial console.

Known Issues

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

  Reducing the TX power in the Advanced Settings 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.

CHAPTER 34

Gluon 2016.2.5

This version contains only a single bugfix for a regression introduced in Gluon v2016.2.4. As the regression affects batman-adv compat 15 only, firmwares using the compat 14 legacy version don't need to be updated.

Bugfixes

- Fix kernel crash with batman-adv compat 15 (d452a7c)

  An incorrect backport of a fix for a very improbable kernel crash caused a much more frequent kernel crash. The backport has been fixed.

  This bug a regression in Gluon v2016.2.4.

Known Issues

- x86 sysupgrade (sometimes) loses config when kernel partition grows (#1010)

  This issue affects upgrades from v2016.2.x and older to the Gluon master only, we hope to fix it before the next major release.

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

  Reducing the TX power in the Advanced Settings 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.

CHAPTER 35

Gluon 2016.2.4

Bugfixes

- Fix batman-adv (compat 15) not being able to transmit packages of specific sizes (b7eeef9)

  We suspect that this issue was also the reason for the autoupdater/wget hangs observed by many communities. Non-Gluon nodes like gateways should be updated to batman-adv 2017.0.1 to get the fix.

- Fix build after ftp.all.kernel.org discontinuation (#1059)
- Fix high load because of frequent calls of the respondd initscript (9a0aeb9)

  The respondd restart triggers added in v2016.2.3 ran a significant portion of the respondd initscript for each router advertisement received. This was fixed by a backport of a netifd patch.

- x86 sysupgrade fixes (41fd50d, ad37e2b)

  This fixes sysupgrade on mmcblk and similar devices.

Other changes

- The manifest generator has been extended to generate SHA256 checksums in addition to SHA512 ones (f9d59be)

  We have recently switched the autoupdater to SHA256 in the Gluon master to avoid mixing two different lengths of hashes for no good reason. This makes the manifests of Gluon v2016.2.x compatible with the new autoupdater so it doesn't prevent backports or downgrades.

  Note: Downgrades of major Gluon versions are generally unsupported and will often lead to broken configurations.

Known Issues

- x86 sysupgrade (sometimes) loses config when kernel partition grows (#1010)

  This issue affects upgrades from v2016.2.x and older to the Gluon master only, we hope to fix it before the next major release.

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

  Reducing the TX power in the Advanced Settings 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.

CHAPTER 36

Gluon 2016.2.3

Added hardware support

ar71xx-generic

- TP-Link TL-WR940N v4
- TP-Link TL-WR1043ND v4

Removed hardware support

Support for Meraki devices (MR12/16/62/66) has been removed for now because of severe problems (all devices were using the same MAC addresses). Support will return when the issues have been fixed.

Bugfixes

- Automatically restart respondd on failure (#863)

  There have been many reports of respondd processes disappearing; the exact cause is unclear, but might be related to the batman-adv debugfs interface and/or out-of-memory conditions.

  A new respondd initscript uses procd to automatically restart respondd when it dies.

- Make autoupdater timeouts more robust (#987)

  It was reported that wget processes sometimes hang indefinitely during the autoupdater manifest download. The autoupdater has been improved to ensure that wget can always be interrupted after a timeout.

  This issue, together with the recent addition of lock files to ensure that only one instance of the autoupdater can run at a time, had caused the autoupdater to blocked completely by hanging processes in some cases (till a node was rebooted).

- Fix regulation domain switching in ath10k (#1001)

  Prevents use of too high transmission power in some cases.

- Ensure that *prefix6* in site.conf is always a /64 prefix (6b62e2f)

  Other prefix lengths were never supported and don't make sense in many places the prefix is used. Ensure that such configurations will not pass validation.

Known Issues

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

  Reducing the TX power in the Advanced Settings 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.

CHAPTER 37

Gluon 2016.2.2

Added hardware support

ar71xx-generic

- TP-Link
  - CPE210/510 EU/US versions
  - TL-WA801N/ND v3
  - TL-WR841ND v11 EU/US versions

Bugfixes

- Fix boot on certain QCA955x-based devices (e.g. OpenMesh OM5P AC v2) (#965)

  This issue was a regression in Gluon v2016.2.1.

- Build: Fix git downloads from git.kernel.org on Debian Wheezy (#919)

  We've switched back from HTTPS to the git protocol for now to avoid using the old GnuTLS version of Debian Wheezy which can't establish a HTTPS connection with git.kernel.org anymore.

  This issue was a regression in Gluon v2016.2.

- Fix RX filter of Ubiquiti UAP Outdoor+ (d43147a8e03d)

  This issue was a regression in Gluon v2016.2.

- Fix switched WAN/LAN interface assignment on CPE210 (59deb2064d54)

  This issue was a regression in Gluon v2016.2.

- Significantly reduce CPU load used by signal strength LEDs (#897)
- Fix ethernet port of the Ubiquiti UAP AC Lite (#911)

- Build: Don't use host `/tmp` directory (f9072a36411b)

  Fixes build when `/tmp` is mounted with *noexec*.

- Fix mesh interface type respondd/alfred announcements when using VLANs over IBSS (#941)
- Fix next-node ebtables rules without *next_node.ip4* (9dbe9f785d2b)

  Gluon v2016.2 added support for using the next-node feature without specifying an IPv4 address. Some scripts had not been adjusted, making the next-node unreliable when no IPv4 address was specified.

Other changes

- x86-generic and x86-64 images now have PATA and MMC support to allow using them on various devices that were previously unsupported.
- Clean up opkg postinst scripts up on image generation

  OpenWrt does this by default to save a little space.

Known Issues

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

  Reducing the TX power in the Advanced Settings 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.

CHAPTER 38

Gluon 2016.2.1

Added hardware support

ar71xx-generic

- TP-Link
  - TL-WA901ND v4

Bugfixes

- Make status page work with disabled cookies/local storage (#912)
- Update kernel to 3.18.44

  Fixes CVE-2016-5195 and CVE-2016-7117. It is unlikely that these issues pose a threat to usual Gluon setups, but installing additional packages may make a system vulnerable. In any case, updating is highly recommended.
- Downgrade mac80211 to an earlier state

  Unfortunately, a mac80211 update that was done shortly before the release of Gluon v2016.2 (that seemed necessary to properly support ath10k devices) had again caused severe ath9k stability issues that remained unreported until v2016.2 was out.

  We have now reverted mac80211 to an earlier state that was reported to be very stable (while keeping the ath10k-specific changes); in addition, some patches that were reported to cause connection or performance issues with certain clients have been reverted. While is it still not perfectly stable, is should be at least as good as (and probably better than) the v2016.1.x release series.

Known Issues

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

Reducing the TX power in the Advanced Settings 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.

- Git HTTPS downloads from git.kernel.org fail on Debian Wheezy (#919)

  The GnuTLS version on Debian Wheezy is too old and can't establish a connection with git.kernel.org anymore. A newer GnuTLS version is available in wheezy-backports, but as there is no libcurl3-gnutls package linked against the new version, installing the new version has no effect.

CHAPTER 39

Gluon 2016.2

Added hardware support

ar71xx-generic

- ALFA Network
  - Tube2H
  - N2
  - N5
- Buffalo
  - WZR-HP-G300NH2
- GL Innovations
  - GL-AR150
- OpenMesh
  - MR1750 v1, v2[1]
  - OM2P-HS v3
  - OM5P-AC v1, v2[1]
- TP-Link
  - Archer C5 v1[1]
  - Archer C7 v2[1]
  - TL-WR710N v2.1
  - TL-WR842N/ND v3

[1] Device uses the ath10k WLAN driver; no image is built unless GLUON_ATH10K_MESH is set as described in *Make variables*

- Ubiquiti
  - UniFi AP AC Lite[1]
  - UniFi AP AC Pro[1]

brcm2708-bcm2708

- RaspberryPi 1

brcm2708-bcm2709

- RaspberryPi 2

New features

- Many UBNT Airmax XM model names are detected correctly now (e.g., the Loco is no longer displayed as Bullet) (#632)

  Also, various new image aliases have been added for these devices.

- batman-adv: mesh_no_rebroadcast is now enabled for Mesh-on-WAN/LAN (#652)
- The new UCI option `gluon-core.@wireless[0].preserve_channels` can be used to prevent a changed WLAN channel from being reset on firmware upgrades (#640)
- PoE passthrough can now be configured from site.conf and the Advanced Settings on TP-Link CPE 210/510 and Ubiquiti NanoStations (#328)
- The config mode *altitude* field can now be hidden using the `config_mode.geo_location.show_altitude` site.conf setting (#693)
- The contact information field in the config mode can be made obligatory using the `config_mode.owner.obligatory` site.conf option
- The *node name* setting in the config mode is no longer restricted to valid DNS hostnames, but allows any UTF-8 string (#414)
- Besides the hostname, public key, site config and primary MAC address, the contact information can now be accessed from config mode site texts
- The functions `escape` and `urlescape` for HTML and URL escaping are now available from config mode site texts. They should always be used when including user-provided information like hostnames and contact information in HTML code or URLs.
- Dropbear has been updated to a newer version, enabling new SSH crypto methods and removing some old ones like DSA. This reduces the time needed for the first boot and makes SSH logins faster (#223)
- WLAN basic and supported rate sets have been made configurable, to allow disabling 802.11b rates (#810)
- ath10k-based devices are now supported officially; it's possible to choose between IBSS- and 11s-capable firmwares in site.mk (#864)
- The `prefix4` and `next_node.ip4` site.conf options are optional now.

Bugfixes

- The stability of the ath9k WLAN driver has been improved significantly (#605)

  mac80211, hostapd and other related drivers and services have been backported from LEDE `42f559e`.
- Extremely slow downloads could lead to multiple instances of the autoupdater running concurrently (#582)

  A lockfile is used to prevent this and timeouts have been added to download processes.
- Usage of static DNS servers on the WAN port has been fixed (#886)

  This is a regression introduced in Gluon v2016.1.6.

Other changes

- The "Expert Mode" has been renamed to "Advanced Settings"

Site changes

site.mk

If you want to support ath10k-based devices, you should set GLUON_ATH10K_MESH and GLUON_REGION as described in *Make variables*.

i18n

As the hostname field may now contain an arbitrary UTF-8 string, escaping must be added.

Change

```
<%=hostname%>
```

to

```
<%=escape(hostname)%>
```

Inside of URLs, `urlescape` must be used instead of `escape`.

Internals

- Mesh interfaces are now configured in a protocol-independent way in UCI (#870)

  The MAC address assignment of all mesh and WLAN interfaces has been modified to prepare for support of Ralink/Mediatek-based WLAN chips.
- Preparations for supporting the new batman-adv multicast optimizations have been made (#674, #675, #679)
- All Lua code is minified now to save some space

Known Issues

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

  Reducing the TX power in the Advanced Settings 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.

CHAPTER 40

Gluon 2016.1.6

Bugfixes

- build: fix nodejs host build on Debian Wheezy (#776)
- build: fix parallel builds with Make 4.2+

  Trying to use `-j N` with Make 4.2 would spawn an unlimited number of processes, eventually leading to memory exhaustion.
- build: fix occasional build failure in libpcap package
- build: don't require hexdump for x86 builds (#811)

  Trying to build Gluon for x86 on systems without hexdump would silently generate broken images.
- Add support for DNS servers given by their link-local IPv6 address in Router Advertisements (#854)
- ar71xx-generic: correctly setup LNA GPIOs on CPE210/510 (#796)

  Improves the reception by about 20dB.
- ar71xx-generic: switch default WAN/LAN assignment on Ubiquiti UAP Pro (#764)

  Switch to the usual "PoE is WAN/setup mode, secondary is LAN" scheme. This only affects new installations; the assignment won't be changed on updates unless the configuration is reset.
- ar71xx-generic: fix ath10k memory leak (#690)
- ar71xx-generic: add support for new TP-Link region codes (#860)

  TP-Link has started providing US- and EU-specific firmwares for the Archer C7 v2. To generate Gluon images installable from these new firmwares, the `GLUON_REGION` variable must be set to `eu` or `us` in `site.mk` or on the `make` command line (the images will still be installable from all old firmwares without region codes).

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.

CHAPTER 41

Gluon 2016.1.5

Added hardware support

ar71xx-generic

- OpenMesh
- MR600 (v1, v2)
- MR900 (v1, v2)
- OM2P (v1, v2)
- OM2P-HS (v1, v2)
- OM2P-LC
- OM5P
- OM5P-AN
- Ubiquiti
- Rocket M XW
- TP-LINK
- TL-WR841N/ND v11

Bugfixes

- build: fix race condition caused by using certain make targets (like *clean*, *images* or *package/**) with parallel build options without doing a full build before

- build: fix package dependency issue causing “recursive dependency” warning

  This dependency issue could lead to broken configurations and reportedly caused failed builds in some cases when additional (site-specific) packages were used.

- build: Gluon will now build correctly with GCC 6 as host compiler
- Fix configuration of batman-adv when VLANs are used on top of IBSS interfaces (regression due to netifd update in *Gluon 2016.1.4*)
- Add back missing ath10k firmware (regression due to mac80211 update in *Gluon 2016.1.4*)
- Gluon can now be used on all supported Ubiquiti AirMAX devices without downgrading to AirOS 5.5.x before

  Gluon 2016.1.1 added support for most Ubiquiti AirMAX devices with AirOS 5.6.x without downgrading AirOS, but left some devices (at least some PicoStations and Bullets) with unwritable flash. This issue has been resolved (#687).

- Add upgrade script to automatically remove whitespace from configured geolocation

  The new respondd implementation included in *Gluon 2016.1* is stricter about the number format than the old one and doesn’t accept trailing whitespace (so one or both coordinates are missing from the output).

  The Config Mode has been fixed to strip whitespace from numeric fields in new configurations since *Gluon 2016.1.1*. This still left old configurations, which are now fixed by this script.

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.

CHAPTER 42

Gluon 2016.1.4

Added hardware support

ar71xx-generic

- 8devices Carambola 2
- Meraki MR12/MR62/MR16/MR66

Bugfixes

- Major update of all WLAN drivers

  We've taken the unusual step of updating the WLAN drivers ("wireless-backports") to a much newer version, as it was reported that the new version fixes unstable WLAN seen in many setups
- Build fix: a race condition causing parallel builds to fail has been fixed
- Build fix: the Gluon tree could get into a state in which all commands fail with "Too many levels of symbolic links"
- Build fix: allow building Gluon on systems with certain versions of *dash* as */bin/sh*

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.

CHAPTER 43

Gluon 2016.1.3

Added hardware support

ar71xx-generic

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

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.

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.

CHAPTER 44

Gluon 2016.1.2

Added hardware support

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

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.

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.

CHAPTER 45

Gluon 2016.1.1

Added hardware support

ar71xx-generic

- Onion Omega
- TP-Link TL-MR13U v1

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.

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.

CHAPTER 46

Gluon 2016.1

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.

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.

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.

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_egress = 200,
    limit_ingress = 3000,
  },
}
```

needs to be changed to

```
fastd_mesh_vpn = {
  -- ...

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

– 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>"
```

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 -> microndd (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*.

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.

CHAPTER 47

Gluon 2015.1.2

Added hardware support

ar71xx-generic

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

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

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

CHAPTER 48

Gluon 2015.1.1

Added hardware support

ar71xx-generic

- TP-Link
  - TL-WA830RE (v1)

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.

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)

CHAPTER 49

Gluon 2015.1

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

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.

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.

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.

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.

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.

CHAPTER 50

Gluon 2014.4

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)

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.

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)

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`!)

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*

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.

CHAPTER 51

Gluon 2014.3.1

This is a bugfix release.

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

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.

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.

CHAPTER 52

Gluon 2014.3

New hardware support

- Linksys WRT160NL

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.

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`.

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.

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 53

Supported Devices & Architectures

ar71xx-generic

- 8devices
  - Carambola 2
- ALFA Network
  - AP121
  - AP121U
  - Hornet-UB
  - Tube2H
  - N2
  - N5
- Allnet
  - ALL0315N
- Buffalo
  - WZR-HP-AG300H / WZR-600DHP
  - WZR-HP-G300NH
  - WZR-HP-G300NH2
  - WZR-HP-G450H
- D-Link
  - DIR-505 (A1, A2)
  - DIR-615 (C1)
  - DIR-825 (B1)

- GL Innovations
  - GL-AR150
  - GL-iNet 6408A (v1)
  - GL-iNet 6416A (v1)
- Linksys
  - WRT160NL
- Netgear
  - WNDR3700 (v1, v2)
  - WNDR3800
  - WNDRMAC (v2)
- Onion
  - Omega
- OpenMesh
  - MR600 (v1, v2)
  - MR900 (v1, v2)
  - MR1750 (v1, v2)[1]
  - OM2P (v1, v2)
  - OM2P-HS (v1, v2, v3)
  - OM2P-LC
  - OM5P
  - OM5P-AN
  - OM5P-AC (v1, v2)[1]
- TP-Link
  - Archer C5 (v1)[1]
  - Archer C7 (v2)[1]
  - CPE210 (v1.0, v1.1)
  - CPE220 (v1.1)
  - CPE510 (v1.0, v1.1)
  - CPE520 (v1.1)
  - RE450
  - TL-WDR3500 (v1)
  - TL-WDR3600 (v1)
  - TL-WDR4300 (v1)
  - TL-WR710N (v1, v2.1)
  - TL-WR842N/ND (v1, v2, v3)

[1] Device uses the ath10k WLAN driver; no image is built unless GLUON_ATH10K_MESH is set as described in *Make variables*

  - TL-WR1043N/ND (v1, v2, v3, v4)
  - TL-WR2543N/ND (v1)
  - WBS210 (v1.20)
  - WBS510 (v1.20)
- Ubiquiti
  - Air Gateway
  - Air Gateway LR
  - Air Gateway PRO
  - Air Router
  - Bullet M2/M5
  - Loco M2/M5
  - Loco M2/M5 XW
  - Nanostation M2/M5
  - Nanostation M2/M5 XW
  - Picostation M2/M5
  - Rocket M2/M5
  - Rocket M2/M5 Ti
  - Rocket M2/M5 XW
  - UniFi AP
  - UniFi AP AC Lite[1]
  - UniFi AP AC Pro[1]
  - UniFi AP LR
  - UniFi AP Pro
  - UniFi AP Outdoor
  - UniFi AP Outdoor+
- Western Digital
  - My Net N600
  - My Net N750

ar71xx-nand

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

ar71xx-tiny

- D-Link
  - DIR-615 (C1)
- TP-Link
  - TL-MR13U (v1)
  - TL-MR3020 (v1)
  - TL-MR3040 (v1, v2)
  - TL-MR3220 (v1, v2)
  - TL-MR3420 (v1, v2)
  - TL-WA701N/ND (v1, v2)
  - TL-WA730RE (v1)
  - TL-WA750RE (v1)
  - TL-WA801N/ND (v1, v2, v3)
  - TL-WA830RE (v1, v2)
  - TL-WA850RE (v1)
  - TL-WA860RE (v1)
  - TL-WA901N/ND (v1, v2, v3, v4)
  - TL-WA7210N (v2)
  - TL-WA7510N (v1)
  - TL-WR703N (v1)
  - TL-WR710N (v1, v2, v2.1)
  - TL-WR740N (v1, v3, v4, v5)
  - TL-WR741N/ND (v1, v2, v4, v5)
  - TL-WR743N/ND (v1, v2)
  - TL-WR841N/ND (v3, v5, v7, v8, v9, v10, v11, v12)
  - TL-WR843N/ND (v1)
  - TL-WR940N (v1, v2, v3, v4)
  - TL-WR941ND (v2, v3, v4, v5, v6)

brcm2708-bcm2708

- RaspberryPi 1

brcm2708-bcm2709

- RaspberryPi 2

mpc85xx-generic

- TP-Link
  - TL-WDR4900 (v1)

x86-generic

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

See also: *x86 support*

x86-geode

- x86-geode

See also: *x86 support*

x86-64

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

See also: *x86 support*

CHAPTER 54

License

See LICENCE

CHAPTER 55

Indices and tables

- genindex
- search