Configuring the in-world representation of system state under Homenet3D for OpenWRT v0.2

Dominic Allan, Grenville Armitage
Centre for Advanced Internet Architectures, Technical Report 140702B
Swinburne University of Technology
Melbourne, Australia
dallan@swin.edu.au, garmitage@swin.edu.au

***Abstract*—‘Homenet3D for OpenWRT’ v0.2 presents the internal state of an OpenWRT router in a 3D graphical form. This report details the structure and functionality of the configuration file that drives in-world representation of system state. We list the attributes and metrics that may be configured and explain how these properties map to the Homenet3D world’s entities. Current limitations in v0.2 are also discussed.**

***Index Terms*—CAIA, Homenet3D, libconfig, OBJ/MTL, L3DGEWorld**

I. Introduction

The Homenet3D [1] project aims to provide a virtual 3D world to graphically display the state of a home network. Ultimately we envisage every router displaying its network and system state using objects rendered and animated in 3D space within a browser window. These objects vary over time in a variety of ways (such as shape, size and colour) in an effort to communicate what is happening on the network.

‘Homenet3D for OpenWRT’ v0.2 is our current prototype targetting the OpenWRT home gateway platform [2]. By default, the Homenet3D server utilises a configuration file located at `/usr/local/share/homenet3d/homenet3d.cfg`. to describe how different 3D objects are mapped to various internal OpenWRT system state. The config file itself is parsed using the `libconfig` C library [3], making it easy to customise attributes in a flexible manner.

The rest of this report is structured as follows. Section II briefly describes the high-level configuration file structure. Section III describes the various attributes and their visual representations within 3D space. Section IV describes how network metrics are mapped to particular attributes. The report concludes in Section V.

II. Config File Structure

The main structure of the config file follows that of a libconfig file structure [4]. Algorithm 1 outlines the basic structure of the Homenet3D config file.

The structure is hierarchical. When the entities are initialised the server program will read through the config file. If it finds a global attribute configuration then that will be assigned to that entity’s attribute. Otherwise it will look for an attribute configuration in the entity itself. If neither exists it will be set to a default, usually 0 or null, depending on the datatype.

Algorithm 1 Homenet3D configuration file structure

```
name = "Homenet3D"
version = "0.2"

application =
{
   /* global configurations: attributes
       and metrics */

   entities = (
      {
         /* configurations for an entity */
      },
      {
         /* configurations for another
             entity */
      }
   )
}
```

III. Attributes

The objects attributes are communicated on a per entity basis. Each entity has all the attributes listed in Table I as well as entity-specific attributes that are defined by the stateType. The entity-specific attributes are used to store information about the system and it's state.

Algorithm 2 gives an example of a basic entity entry in the config file. The configuration has defined the entity's location in the virtual world at point (−180,−70,0), the entity's text label (Wireless), what type of information the entity will display ("wifi" information) and the shape it will be (in this case a star, see Section III-C).

Algorithm 2 Basic entity configuration entry

```
...
  entities = (
    {
      x = -180;
      y = -70;
      z = 0;
      label = "Wireless";
      stateType = "wifi";
      shape = 4;
    },
    ...
  )
...
```

A. *The* stateType

Figure 1 shows the categories of OpenWRT system state that are tracked in v0.2.

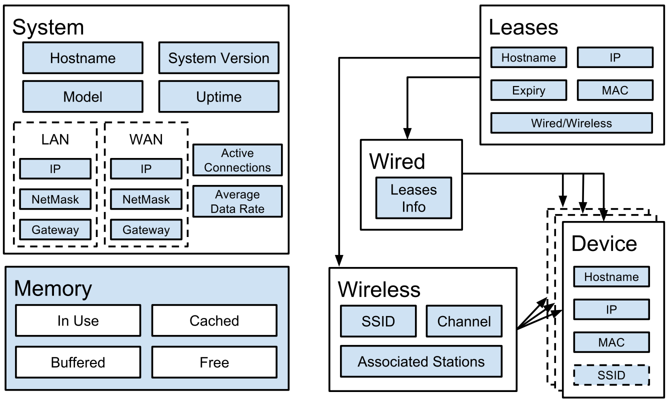


Figure 1. OpenWRT System States tracked in v0.2

We use the stateType identifier to map specific internal states to on-screen entities. At the current version the nature of these state types is static and set at a source code level. The current state types are:

Table I
ATTRIBUTES

Attribute	Format	Description	Config Level
stateType	String	The name for the type of entity the object will represent (Section III-A).	Entity-level
position	vector	The position of an object in 3D space is set by x,y,z (Section III-B).	Entity-level
bounce height	int	The height at which the object will bounce through.	Entity-level Global-level
bounce freq	int	How long it takes for one bounce to occur	Entity-level Global-level
rotate speed	float	The rate at which an object will rotate through X and Y axes	Entity-level Global-level
colour	colour	The colour of the object specified in RGB form (Section III-D).	Entity-level Global-level
text colour	colour	The colour of the object's 3D text label specified in RGB form (Section III-D).	Entity-level Global-level
label	String	The text that is displayed in 3D with the object.	Entity-level
radius	int	The radius of the object. (Works with all shape types)	Entity-level Global-level
scale	float	A ratio of the original size to a new size.	Entity-level Global-level
shape	int	An integer that represents a shape (Section III-C).	Entity-level
rotation	vector	The angle for each axes that the object is rotated (Section III-B).	Entity-level Global-level

- system – system name, system type, model name, uptime
- memcached, memfree, membuffers, memtotal – memory cached, free, for buffers and total memory in use
- network – LAN/WAN (IP address, network mask, gateway), active connections, current data rate on WAN interface (received/transmitted)
- dhcp – all valid leases (IP, MAC, hostname, expiry, client ID for each lease)
- wifi – all the active wireless radios (SSID, channel and associated devices for each radio)
- wired – all devices that have connected to the network via a cable

The config file allows the shape of an entity to be defined by a OBJ/MTL model (Section III-C). For that to work, the id must contain the following state type:

- model – OBJ and MTL file information

These state type can be stacked. One entity can contain multiple state types. For example in the current version of Homenet3D (v0.2) the entity labelled as “System” has both the system and network state types in it’s stateType configuration and so it contains the information from both those state types (shown in Figure 2). The configuration to combine two state types into one entity is shown in Algorithm 3.

Algorithm 3 Dual *stateTypes*

```
...
entities = (
  ...
  {  ...
    stateType = "system network";
  },
  ...
)
```

B. Vectors

The position and the rotation (orientation) of an object are configure using a three dimensional vector. The dimensions are represented by x,y,z. For the position each of these are integers that specify the objects distance on each axis from an arbitrary centre point. From the default camera position the axis are thus, where the centre of each axis is 0:

- **x axis** – the “width”, left(-) to right(+).
- **y axis** – the “height”, bottom(-) to top(+).
- **z axis** – the “depth”, backward(-) to forward(+).

The communication protocol for Homenet3D has been adapted from a L3DGEWorld L3DGEComms protocol [5]. A legacy from this protocol is that the position is specified as three separate attribute: x, y and z.

The rotation of an object is done using radians. That is, for an object to be rotated along the x axis 90° would become $\frac{\pi}{2}$ (see Algorithm 4). The libconfig library does not have a built-in math library and so $\frac{\pi}{2}$ needs to be calculated to a rounded decimal number (e.g. 1.57079). For rotation, if you want to only rotate one axis then you only need to specific that axis. You can rotate any combination of the three axes at the same time in the vector format.

Algorithm 4 Entity rotation

```
...
entities = (
  ...
  {
    ...
    rotation =
    {
      x = 1.57079;
    };

    /* OR */
    rotation =
    {
      x = 1.57079;
      y = 0;
      z = 0;
    };
  },
  ...
)
```

C. Shapes

An entity’s shape can be selected from a number of in built shapes (cube, sphere, tetrahedron[1], cylinder, star, plane[2]) or externally defined OBJ/MTL files for a fully customised object. Table II lists the numeric values used to select a particular shape.

Table II
SHAPES

Number	Shape
1	cube
2	sphere
3	tetrahedron
4	star
5	cylinder
6	plane
7	OBJ/MTL model

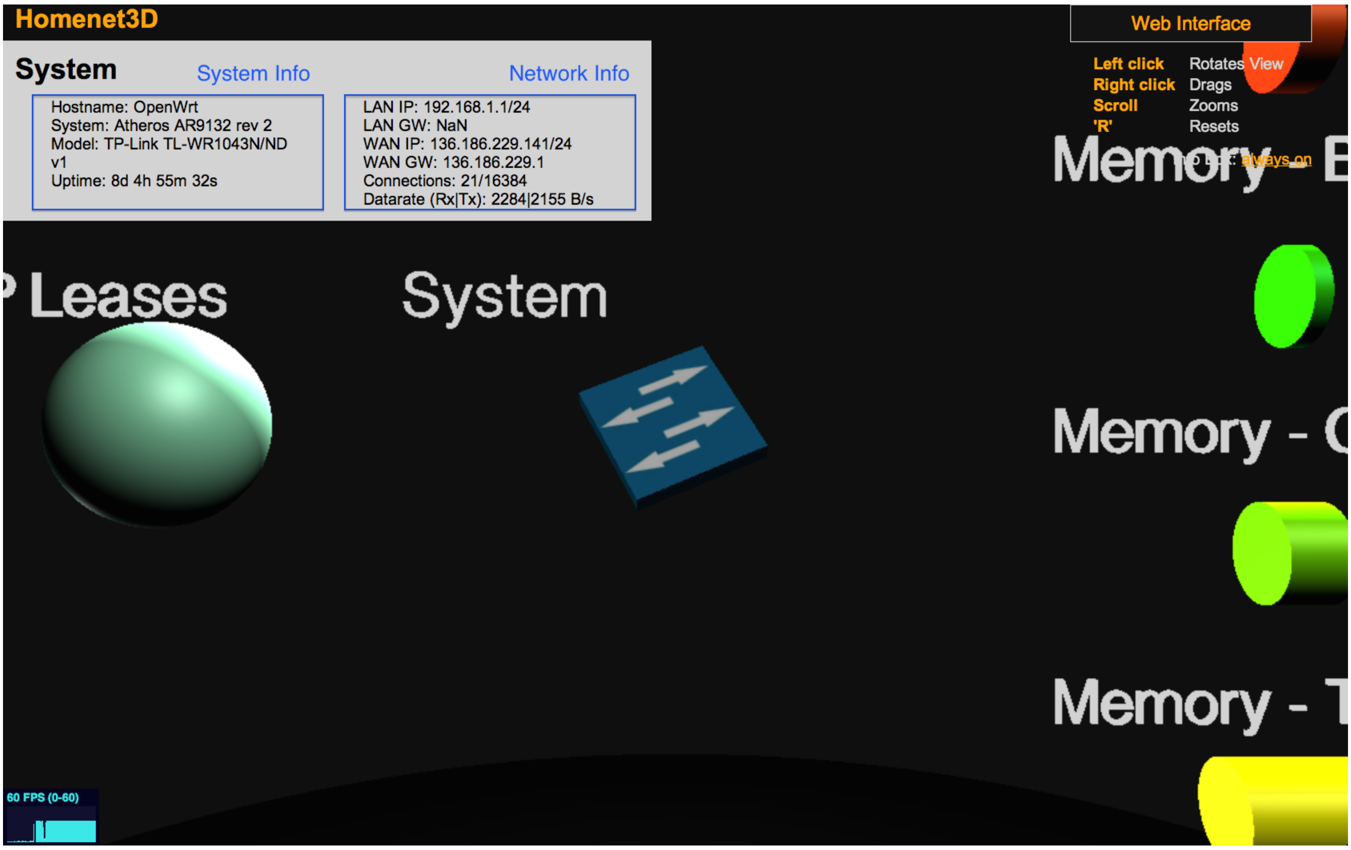
Algorithm 5 illustrates the information required to specify an OBJ/MTL model [6] in the configuration file.[3] There are three key steps:

1) Add the “model” keyword to the stateType.
2) Set the entity’s ‘shape’ to 7.

[1] The tetrahedron implementation is buggy and not recommended.

[2] Actually implemented as a 2D rectangle.

[3] In this case the example refers to a model of a “Cisco switch” icon (used here to represent the core OpenWRT system state).



Hostname: OpenWrt
System: Atheros AR9132 rev 2
Model: TP-Link TL-WR1043N/ND v1
Uptime: 8d 4h 55m 32s

LAN IP: 192.168.1.1/24
LAN GW: NaN
WAN IP: 136.186.229.141/24
WAN GW: 136.186.229.1
Connections: 21/16384
Datarate (Rx|Tx): 2284|2155 B/s

Figure 2. Dual *stateType* information screenshot

3) Set `objfile` and `mtlfile` to point to the relevant OBJ and MTL files (using either absolute or relative paths).

The size of such custom models are set within their OBJ/MTL files. You may need to define and adjust the entity's `scale` attribute until the object is rendered at an appropriate size inside the Homenet3D world.

D. Colours

Object and text colours are defined using a combination of three integers from 0 to 255 to specify each colour's red, green and blue components. An example of how to set an entity's colour to white and it's text to black is shown in Algorithm 6.

IV. Mapping System State to Visual Attributes

In the long term we plan to allow the mapping of any system or network state to particular visual attributes of their corresponding entities. For v0.2 we allow configuration of how the entity representing the WAN interface (`network`) can have its size scaled to represent the number of active connections, and its spin rate set proportional to the average data rate through the interface.

Algorithm 5 Specifying a custom OBJ/MTL model

```
...
  entities = (
    {
      x = 0;
      y = 0;
      z = 0;
      label = "System";

      /* Note the next four lines */
      stateType = "system network model";
      shape = 7;
      objfile = "models/switch.obj";
      mtlfile = "models/switch.mtl";
    },
   ...
  )
...
```

Algorithm 6 Colour attributes

```
...
entities = (
   ...
   {
      ...
      colour = {
         red = 255;
         green = 255;
         blue = 255;
      };
      text_colour = {
         red = 0;
         green = 0;
         blue = 0;
      };
   },
   ...
)
```

Mapping involves specifying how an input range (the measured state, or *metric*) is modified into an output range (values for an attribute). Currently we support reconfiguring the mapping of the following `network` state metrics:

- `conns` – ratio of active connections to total allowed connections[4], which is mapped to object scale using `scale_metric`.
- `datarate` – average datarate on the WAN interface, which is mapped to object rotation speed using `rotate_speed_metric`.

The `conns` and `datarate` metrics take two further parameters – `upper_thres` and `lower_thres` – which define the upper and lower bounds respectively on system state values.

`scale_metric` and `rotate_speed_metric` then map their source's values into new scale and rotation values defined by the following three attributes:

- `max` – attribute value corresponding to the source metric being equal or greater than `upper_thres`
- `min` – attribute value corresponding to the source metric being equal to or less than `lower_thres`
- `granularity` – size of the discrete steps that the attribute value may take between `min` and `max`

Algorithm 7 illustrates how we map the percentage of active connections (`conns`) to the scale of the object representing network state. In this example, the object's scale will vary in steps of 0.1 from 1.0 (for 100% active connections) down to 0.2 (for anything less than 1% active connections).

Algorithm 7 Mapping metrics to visual attributes

```
...
entities = (
   ...
   {
      ...
      conns = {
         upper_thres = 1.0;
         lower_thres = 0.01;
      };
      scale_metric = {
         max = 1.0;
         min = 0.2;
         granularity = 0.1;
      };
   },
   ...
)
```

A similar syntax is used to define the mapping from `datarate` to `rotate_speed_metric`.

V. Conclusion

This report describes the configuration file used by Homenet3D v0.2. The configuration file is loaded each time the Homenet3D server starts, making it easy to change the look and feels of the objects in the world.

The v0.2 configuration file is an early expression of a more complex and flexible system we plan for future releases of Homenet3D for OpenWRT.

Acknowledgement

This project has been made possible in part by a gift from the Cisco University Research Program Fund, a corporate advised fund of Silicon Valley Community Foundation, for a project titled "Low cost home network monitoring using ‘3D virtual environments’"

References

[1] G. Armitage, “Homenet3D,” Jan 2014. [Online]. Available: http://caia.swin.edu.au/urp/homenet3d

[2] D. Allan, “Homenet3D for OpenWRT,” June 2014. [Online]. Available: http://caia.swin.edu.au/urp/homenet3d/openwrt-v0.2

[3] M. Lindner, “libconfig - C/C++ Configuration File Library,” 2013. [Online]. Available: http://www.hyperrealm.com/main.php?s=libconfig

[4] M. Lindner, “libconfig Manual,” 2013. [Online]. Available: http://www.hyperrealm.com/libconfig/libconfig_manual.html#Configuration-Files

[4] Show as a percentage on the traditional OpenWRT GUI.

[5] L. Parry, "L3DGEWorld 2.3 Input & Output Specifications," Centre for Advanced Internet Architectures, Swinburne University of Technology, Melbourne, Australia, Tech. Rep. 080222C, 22 February 2008. [Online]. Available: http://caia.swin.edu.au/reports/080222C/CAIA-TR-080222C.pdf

[6] Wikipedia, "Wavefront .obj file," June 2014. [Online]. Available: http://en.wikipedia.org/wiki/Wavefront_.obj_file#Material_template_library