**Technical Feasibility Test for Open Mobile Network Testbed**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Topology GUI Environment**

Author

Dai Hoang Tran

Kyung Hee University

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Author** | **Version** | **Date** | **Change** |
| Dai H. Tran | 0.1 | 2013/11/21 | Initial drafted document |
| Dai H. Tran | 0.2 | 2013/11/23 | Add more details about framework and classes |

Table of Contents

[A – Glossary 4](#_Toc373078446)

[I – Introduction 5](#_Toc373078447)

[II – System overview 6](#_Toc373078448)

[III – System functionality 7](#_Toc373078449)

[IV – Development tool/framework 8](#_Toc373078450)

[1 – Get the right Eclipse version 8](#_Toc373078451)

[2 – Install GWT into Eclipse 9](#_Toc373078452)

[3 – Testing and Debugging with Chrome 10](#_Toc373078453)

[3.1 – Getting the GWT plugin 10](#_Toc373078454)

[3.2 – Import project to Eclipse by Maven 11](#_Toc373078455)

[V – MVC Design pattern 12](#_Toc373078456)

[1 – View class 13](#_Toc373078457)

[2 – Controller class 14](#_Toc373078458)

[3 – Model class 15](#_Toc373078459)

[4 – EventBus class 15](#_Toc373078460)

[VI – GUI design overview 16](#_Toc373078461)

[VII – SVG Drawing implementation 17](#_Toc373078462)

[VIII – Client/Server communication using RESTful API 18](#_Toc373078463)

[IX – Current application status 19](#_Toc373078464)

# A – Glossary

|  |  |
| --- | --- |
| HTML5 | Hyper Text Markup Language Version 5, the standard markup language for creating web pages |
| RESTful  API | Representational State Transfer Application Programming Interface, predominant web API design model |
| JSON | JavaScript Objection Notation, an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pair |
| IDE | Integrated Development Environment, a software application for software development |
| Eclipse | A famous, open-source IDE |
| GWT | Google Web Toolkit, an open source set of tools that allows web developers to create and maintain complex JavaScript front-end applications in Java |
| Chrome | A Google web browser product |
| MVC | Model-View-Controller, a software architecture used in software engineering, that has 3 entities: Model, View and Controller |
| SVG | Scalable Vector Graphic, XML-based vector image format for two-dimensional graphics that has support for interactivity and animation |
| Git | A distributed revision control and source code management |

# I – Introduction

Topology GUI environment (TGE) is a web application that helps illustrate the Open Mobile Network Testbed (OMNT) topology. Users can perform various interactions with the GUI to examine the whole network topology.

To make it portable and working across different platforms, topology GUI environment is developed to give output as HTML5 (the new web standard markup language). Here are some of the decision was made by the author to develop this TGE:

* The chosen development tool and framework are Google Web Toolkit (GWT) and Eclipse IDE, they are well integrated to each other. GWT helps developers to use JAVA and compile the source code to JavaScript, thus produce a well interactive web application.
* MVC design pattern is applied to provide fine-grain control over the complexity for GUI application.
* RESTful API is used in communication by client (TGE) and the cloud (OMNT)

# II – System overview

Figure 1 describes the abstract level of how the TGE fits into the whole OMNT. User have a set of user interface (UI) components such as buttons, menus in the TGE to interact with. Base on that, the TGE will use the OMNT cloud Application Programming Interface (API) to get various topology information. The API is implemented as RESTful web API, thus the API will return JavaScript Object Notation (JSON) as its result.

Figure 1: System overview

# III – System functionality

The table below describes the set of initial functionality that user can interact with the TGE. New functionality will be added in futures as the project requirements increase.

|  |  |
| --- | --- |
| **Function** | **Action** |
| 1 – See all OMNT topology information | User clicks on button “get topology” |
| 2 – Zoom into a cluster, get its information | User clicks on “zoom” button, then clicks on a cluster |
| 3 – View a path-flow connection from one end-host/switch (node) to another node in the topology | User clicks on “make path” button, then clicks on two nodes |

Table 1: System functionality

# IV – Development tool/framework

The author uses GWT as the framework for development because of the author's proficiency in Java programming language. The chosen IDE is Eclipse, an open-source, versatile IDE to help speed up the development process. Other reasons for choosing GWT are the rich set of provided widgets (an UI component) and its strong open-source community. When compiling the source code, GWT will convert Java into JavaScript code to run in the browser. And the main target browser in the development is Google Chrome. The details of how to development environment was setup are describes below.

## 1 – Get the right Eclipse version

Eclipse is a mature open-source IDE that support many programming languages, frameworks and has thousands of plugins that support almost any aspect of programming. As a result, Eclipse IDE comes with a variety of versions and distributions. The author has decided to get the standard JAVA distribution named: “Eclipse IDE for Java Developers” that cooperates the latest Eclipse version 4.3, which includes support for Java programming environment, Git version control and many other useful features that support Java developers. Figure 2 shows the chosen Eclipse IDE version from Eclipse download website.



Figure 2: Eclipse IDE Java version

## 2 – Install GWT into Eclipse

To enable the GWT in Eclipse, the author needs to install the GWT plugin that is provided by Google. Fortunately, Eclipse makes the GWT installation relatively easy by using the Eclipse Marketplace (EM). EM is the also a plugin of Eclipse, that acts as a centralize plugin repository and provide one-click installation experience. Developers can search the plugin he/she wants, then click on install, the rest will be handled by Eclipse itself.

To install GWT, simply open Eclipse Marketplace in Eclipse menu Help -> Eclipse Marketplace. Type in the search box GWT and press search, then choose the plugin “Google Plugin for Eclipse 4.3” (Figure 3), click “install” to get the GWT framework.

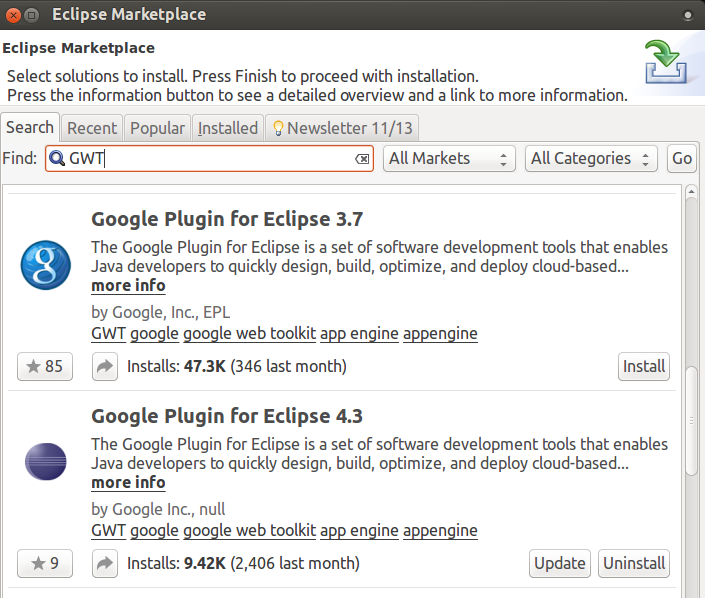


Figure 3: GWT installation in Eclipse

## 3 – Testing and Debugging with Chrome

### 3.1 – Getting the GWT plugin

With the main target browser is Google Chrome, the author performed testing and debugging also with Chrome browser. For this purpose, we need to install specific plugin for Chrome. The plugin name is “GWT Developer Plugin”, it can be easily install at Chrome web store. Figure 4 shows the plugin in Chrome web store.

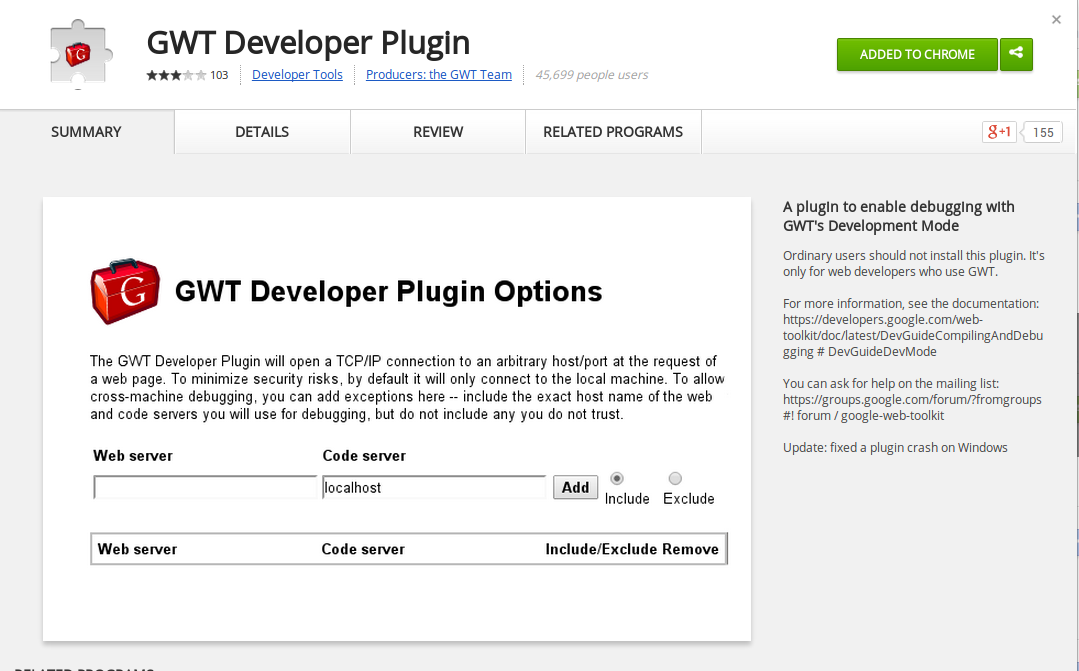


Figure 4: GWT plugin in Chrome web store

### 3.2 – Import project to Eclipse by Maven

TGE project was developed using Maven for project management. Maven is a powerful software project management that can manage project build, reporting and documentation. Maven is well integrated with Eclipse through its plugin, thus new developer can install and import this TGE easily and continue the development cycle. To import TGE using Maven, from Eclipse IDE, choose menu “File” -> “Import” -> click on “Maven” -> “Existing Maven Projects”. Now choose the TGE project folder. Figure 5 shows the process of importing TGE.

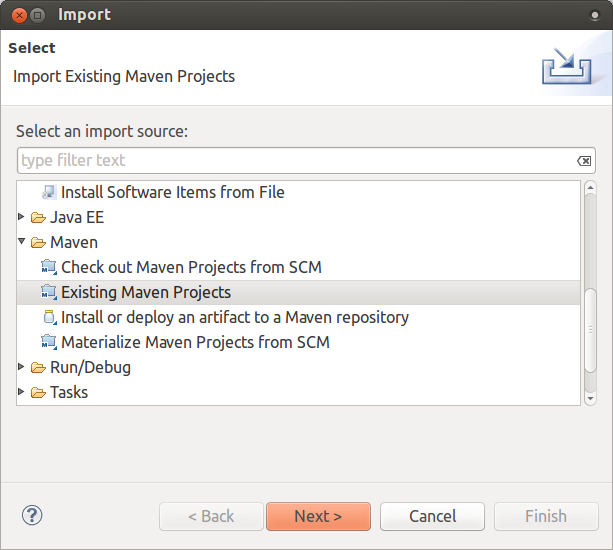


Figure 5: Import Maven project

# V – MVC Design pattern

The TGE application has medium complexity in user interaction. It can evolve to a complex GUI application later on. To address this issue, the author applied a modified Model-View-Controller pattern when implementing the application. Figure 6 shows how the modified MVC works in TGE. Each view will have its own controller. When a user interacts with the view, the events will be sent to a separate component called the EventBus. The EventBus will perform some reasoning to deliver the events to appropriate controllers. This model focuses on centralizing the user event processing part to one entity, which can help organize and avoid messy events handling. The classes are described as follows.

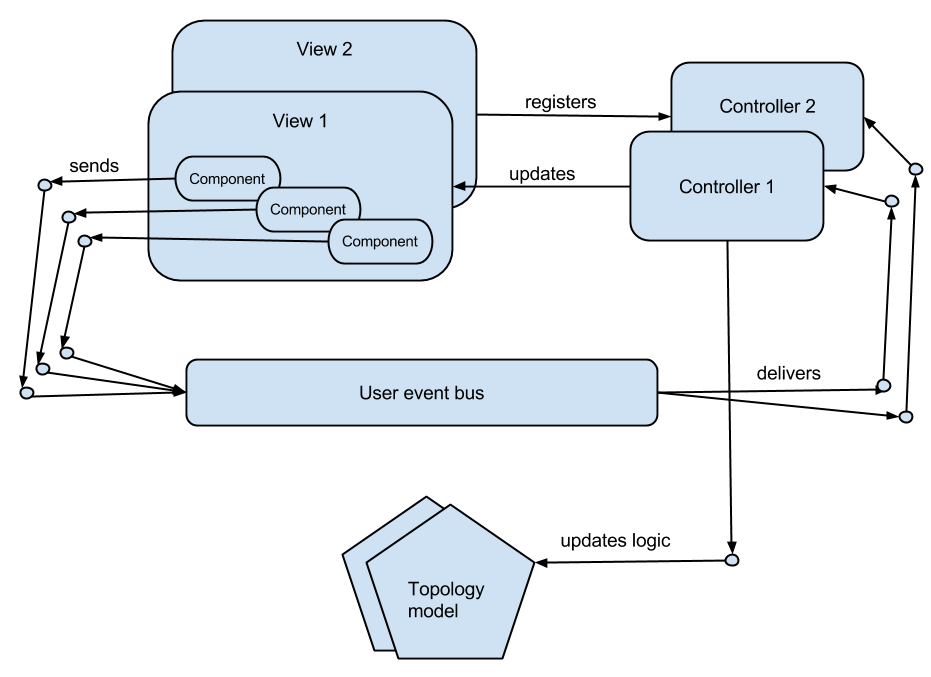


Figure 6: Modified MVC architecture

## 1 – View class

View classes represent the User interface components on the web pages that users can see and interact with. Most of the View classes are extended classes from the GWT widget library. Inside a View class, there can be many child components, which can be seen as sub View classes as well, as illustrated in Figure 9. For example, inside the menu bar, we will have multiple menu buttons, and inside tab bar panel, we can have drawing panel or information panel.

Not all view has controller, for example, standard widget like button or label will not have associate controller. Only the view that has many child component will have controller.

Each view also has its parent that is registered when the view is created from the parent.

Last but not least, each view can update itself to reflect the event interaction. For example, the drawing panel will display all the network topology after user click on show “Global topology” button. Figure 7 shows the common fields/methods of a View class.

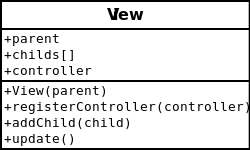


Figure 7: General View class

## 2 – Controller class

Controller classes is the mediate entity that connect View and business Model of the TGE that handle generate events. It is important to distinguish two type of events in the TGE, they are:

* View events: Events generated by UI components such as button clicked, menu item clicked, etc. All of these event will be forwarded to the View’s controller.
* User events: These events are for user interaction, for example, user clicks on “Global topology” button, not only a button clicked event generated, but also a user event is called. But user events will be sent to EventBus class, not the View’s controller.

Besides handling events, controller class also update business models, which are network topologies. Figure 8 shows a general controller class fields/method.

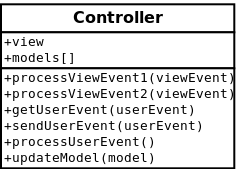


Figure 8: General controller class

## 3 – Model class

Model classes are the classes that represent network topology such as switches, routers, gateways, connections between them, etc. Each of these model will have different attributes, and its information is delivered from the OMNT cloud, which can be requested by calling its RESTful API for its returned JSON result. Currently, there are 4 types of network topology, and are abstracted into 2 types Node and Link:

* Switch: represents a switch router (a node)
* End-host: represents a physical computer that connect to the OMNT (a node)
* Gateway: represents a gateway (a node)
* Link: represents a connection between 2 nodes

## 4 – EventBus class

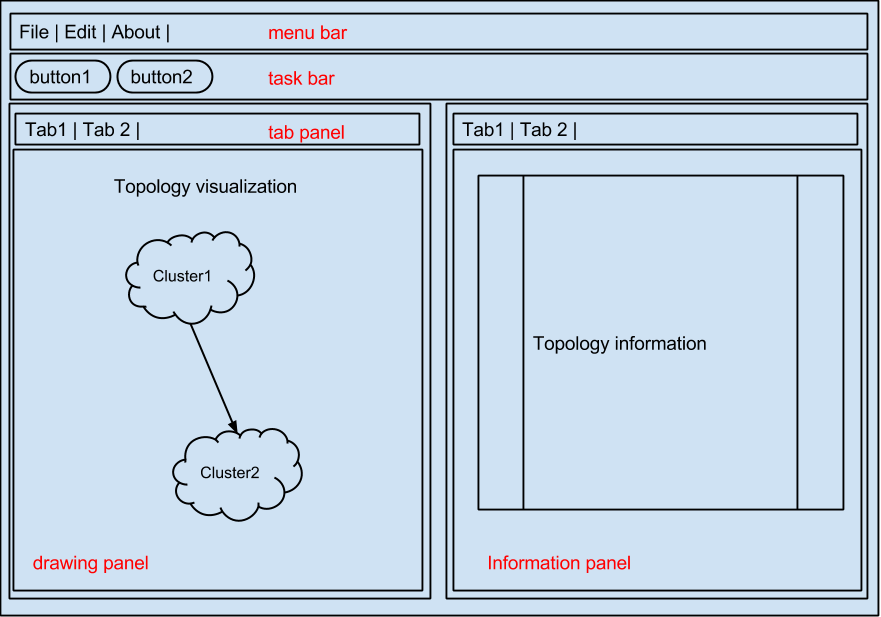
EventBus is an important class that will direct the interaction flow of user and the TGE. The biggest problem with most of the GUI application is that when the application grows bigger with more complexity in user interface, maintaining a correct logic of user events handling can be frustrating. For example, in the TGE, when a user clicks on button “Global topology”, various things happen.

* First the user event will be processed, in which a request is sent to OMNT cloud’s RESTful API to get network topology information.
* When JSON result is returned, TGE will create model classes (Node and Link classes), then updates the drawing panel and information panel for displaying results to user.

As you can see, for just one user event, many class entities get involved, including controllers, views and models. If we don’t have a standard way to control the flow, the program codes will become difficult to understand. As illustrated in Figure 6, the EventBus class will forward user events to appropriate controllers, and these controllers will update appropriate views and models instances.

# VI – GUI design overview

Figure 9 shows the general design of the TGE application. Although it is a web application, the layout is designed to be similar with a normal desktop application including menu bar and task bar, which provides various options to user. For displaying topology information, a left and right panel are used. Both of them can display information in different tabs, whereas left panel will display visualized information including image, drawing of shapes, etc., and the right panel will show detail information about the topology such as node IP address, its connections, etc.

Figure 9: GUI design overview

# VII – SVG Drawing implementation

For visualized drawing of network topology, the author used Scalable Vector Graphic (SVG), an XML-based vector image format for drawing in the browser. SVG has been supported by many major browsers on the web today. The main benefit of using SVG is that the author can modify the image at runtime to dynamically alter the visualization, which makes the TGE more interactive and attractive.

For example, when a user wants to zoom into a cluster, the author can draw all the nodes inside that cluster as an overlay layer on top of a blurry cluster image in the background, which has been scale up and add blurry attribute at the moment user clicks zoom in. This help user to quickly distinguish all the children nodes are belong to that cluster behind. Figure 10 shows an example of this approach.

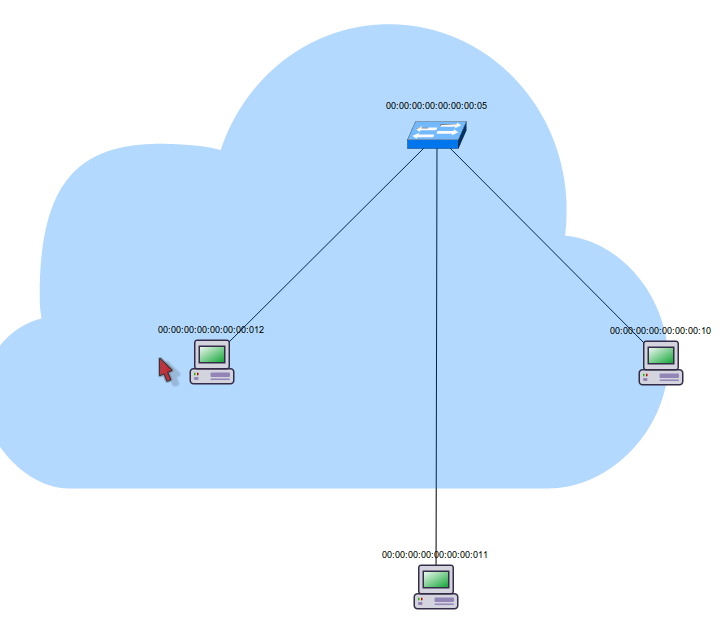
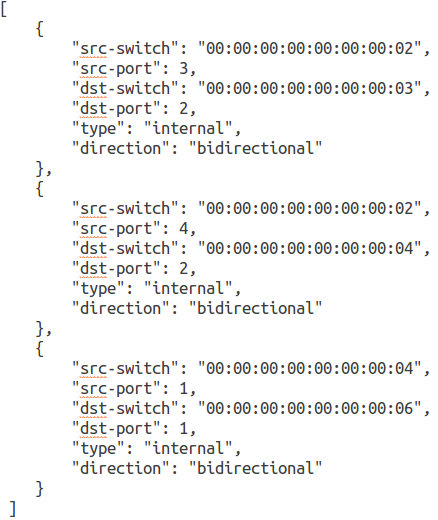


Figure 10: change SVG image at run-time

# VIII – Client/Server communication using RESTful API

When the TGE wants to get network topology information based on user interaction, it will send HTTP requests to the RESTful API service of the OMNT that resided in the cloud. RESTful API help simplify and speed up the communication between the client and the cloud. JSON is the returned result, which is very lightweight. The TGE will quickly process JSON and give user information. Figure 11 shows an example of JSON result

Figure 11: Example JSON result

# IX – Current application status

The TGE is still under development. This session will highlight the current status of the application by showing snapshot of application screen during the development.

Figure 12 shows the application when user gets the whole network topology, and Figure 13 shows the topology inside a cluster (cloud image) when user click “zoom in” button.

On the left side of the TGE, drawing panel will draw network topology by using SVG features of HTML. On the right side, information panel will give user details topology information about each node and link. When user zoom-in into a cluster, new tab will be created to show to nodes inside that cluster (Figure 13).

The next feature that is under development is illustrating the shortest connection path between two nodes when user clicks on “Make path” button.

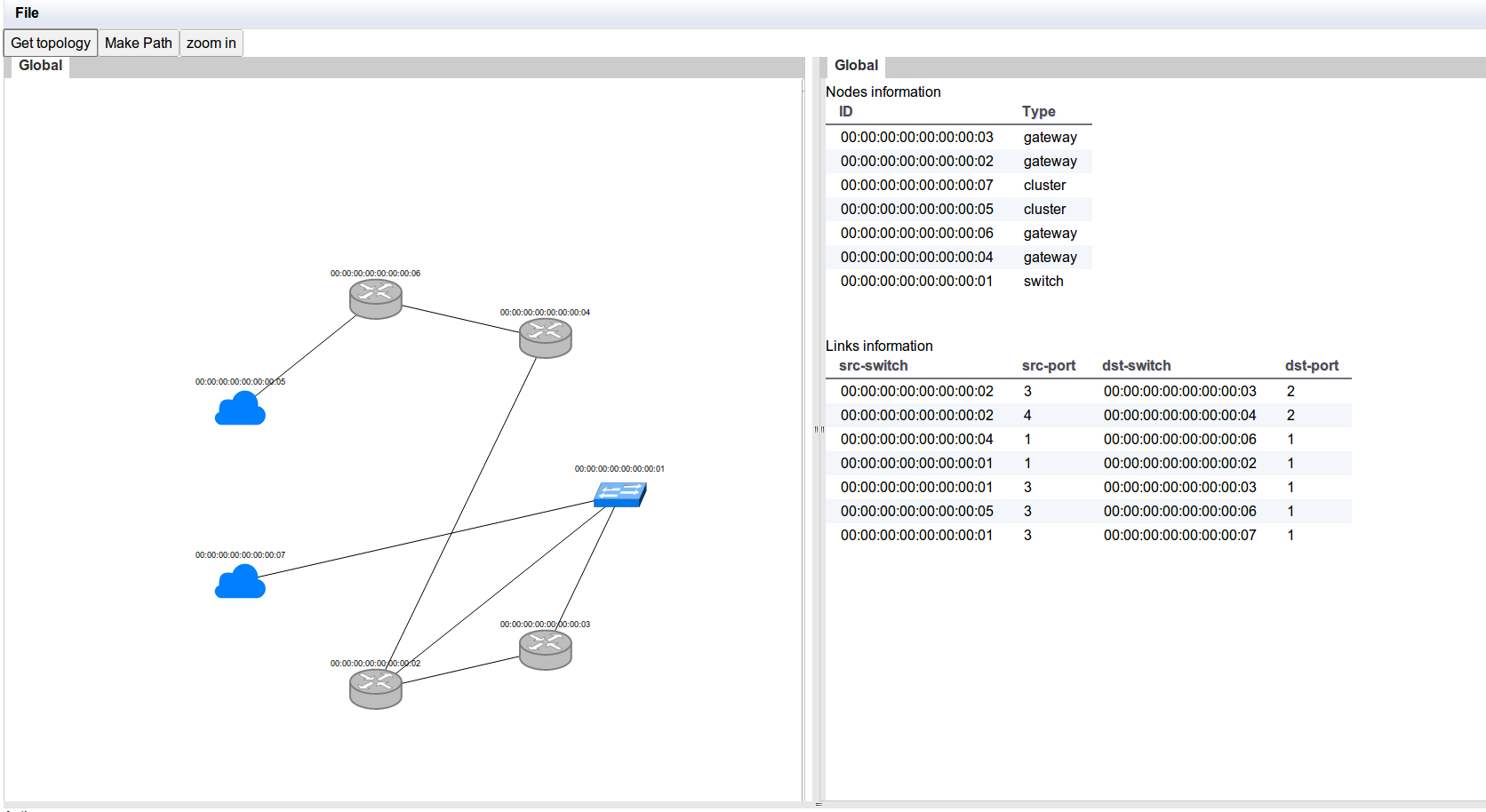


Figure 12: Get global network topology

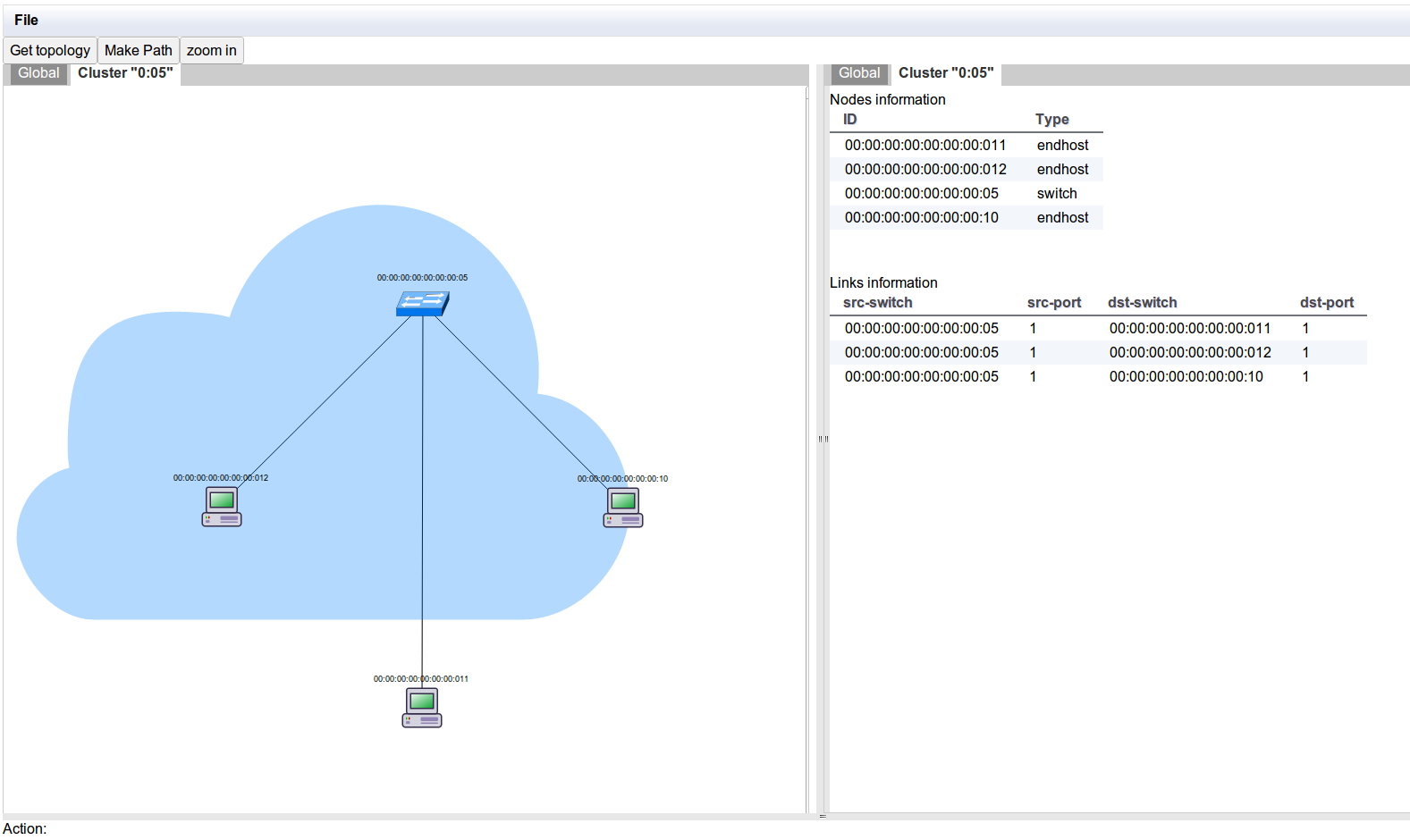


Figure 13: Zoom into a cluster