Project acronym: FORGE
Project full title: "Forging Online Education through FIRE"
Grant agreement no: 610889

D3.1.1 LMS and eBooks widgets for prototype lab courses

Deliverable Editor: Alexander Mikroyannidis (The Open University, UK)

Other Contributors: Christos Tranoris, Daan Pareit, Jono Vanhie-Van Gerwen, Guillaume Jourjon, Pierre Vigueux, Allan Third, John Domingue

Reviewers: Johann M. Marquez Barja, Valina Katsaounou

<table>
<thead>
<tr>
<th>Document Identifier:</th>
<th>FORGE-2014-P-D311</th>
<th>Due Date:</th>
<th>31/07/2014</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version:</td>
<td>1.2</td>
<td>Submission date:</td>
<td>30/07/2014</td>
</tr>
<tr>
<td>State:</td>
<td>Final</td>
<td>Distribution level:</td>
<td>Public</td>
</tr>
</tbody>
</table>

This document is part of a research project funded by FP7 Framework Programme. Grant agreement number 610889.
## Change Log

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Amended by</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>12/06/2014</td>
<td>Alexander Mikroyannidis</td>
<td>Outline and assignment of partner responsibilities</td>
</tr>
<tr>
<td>0.2</td>
<td>27/06/2014</td>
<td>Alexander Mikroyannidis, Christos Tranoris, Daan Pareit, Jono Vanhie-Van Gerwen, Guillaume Jourjon, Pierre Vigreux, Allan Third, John Domingue</td>
<td>Contributions by responsible partners</td>
</tr>
<tr>
<td>0.3</td>
<td>03/07/2014</td>
<td>Alexander Mikroyannidis</td>
<td>Draft for internal review</td>
</tr>
<tr>
<td>0.4</td>
<td>15/07/2014</td>
<td>Johann M. Marquez Barja</td>
<td>Revised draft</td>
</tr>
<tr>
<td>0.5</td>
<td>16/07/2014</td>
<td>Valina Katsaounou</td>
<td>Revised draft</td>
</tr>
<tr>
<td>1.0</td>
<td>25/07/2014</td>
<td>Alexander Mikroyannidis, Christos Tranoris, Guillaume Jourjon, Pierre Vigreux</td>
<td>Finalised draft</td>
</tr>
<tr>
<td>1.1</td>
<td>30/07/2014</td>
<td>Daan Pareit, Jono Vanhie-Van Gerwen</td>
<td>Added some small clarifications</td>
</tr>
<tr>
<td>1.2</td>
<td>30/07/2014</td>
<td>Aneta Tumilowicz</td>
<td>Final QA</td>
</tr>
</tbody>
</table>
# Table of Contents

Change Log .................................................................................................................. 2  
Table of Contents ......................................................................................................... 3  
List of Figures ............................................................................................................. 5  
Executive Summary ..................................................................................................... 7  
1 Overview of technologies ......................................................................................... 8  
   1.1 Definitions ........................................................................................................ 8  
   1.2 The FORGE infrastructure for widgets ......................................................... 9  
2 Prototype widgets .................................................................................................... 12  
   2.1 The ssh2web FORGEBox widget .................................................................. 12  
      2.1.1 Learning context .................................................................................... 14  
      2.1.2 Targeted FIRE testbed .......................................................................... 14  
      2.1.3 Implementation ...................................................................................... 14  
      2.1.4 Further development plans ................................................................... 18  
   2.2 The RemoteGraphViewer FORGEBox widget ............................................ 19  
      2.2.1 Learning context .................................................................................... 19  
      2.2.2 Targeted FIRE testbed .......................................................................... 19  
      2.2.3 Implementation ...................................................................................... 19  
      2.2.4 Further development plans ................................................................... 20  
   2.3 The RemoteLogViewer FORGEBox widget .................................................. 21  
      2.3.1 Learning context .................................................................................... 21  
      2.3.2 Targeted FIRE testbed .......................................................................... 21  
      2.3.3 Implementation ...................................................................................... 22  
      2.3.4 Further development plans ................................................................... 22  
   2.4 The WLAN widget bundle ............................................................................. 23  
      2.4.1 Learning context .................................................................................... 23  
      2.4.2 Targeted FIRE testbed .......................................................................... 23  
      2.4.3 Implementation ...................................................................................... 23  
      2.4.4 Further development plans ................................................................... 34  
   2.5 The LabWiki widget ....................................................................................... 35  
      2.5.1 Learning context .................................................................................... 35  
      2.5.2 Targeted FIRE testbed .......................................................................... 35  
      2.5.3 Implementation ...................................................................................... 36  
      2.5.4 Further development plans ................................................................... 41  
   2.6 The PlanetLab widget ..................................................................................... 42  
      2.6.1 Learning context .................................................................................... 42  
      2.6.2 Targeted FIRE testbed .......................................................................... 42  

---

2014 © Copyright lies with the respective authors and their institutions.
2.6.3 Implementation ................................................................. 43
2.6.4 Further development plans .................................................. 43
2.7 The Wireshark widget .............................................................. 44
  2.7.1 Learning context ............................................................... 44
  2.7.2 Targeted FIRE testbed ....................................................... 45
  2.7.3 Implementation ............................................................... 45
  2.7.4 Further development plans ............................................... 46
2.8 The Packet Tracer widget .......................................................... 47
  2.8.1 Learning context ............................................................... 47
  2.8.2 Targeted FIRE testbed ....................................................... 48
  2.8.3 Implementation ............................................................... 48
  2.8.4 Further development plans ............................................... 49
3 Conclusions and next steps ......................................................... 50
APPENDICES ........................................................................... 51
    Appendix A – Acronyms list ................................................... 51
List of Figures

Figure 1: The FORGEBox birds-eye architectural view .................................................. 10
Figure 2: A still from the FORGE iBook teaser video, demonstrating the use of an HTML5 widget inside the FORGE iBook ............................................................... 11
Figure 3: Accessing FIRE resources via SSH and web-based SSH .................................. 12
Figure 4: The ssh2web FORGEBox widget displayed in the web page of a course ............ 14
Figure 5: Architecture of ssh2web widget ................................................................. 15
Figure 6: Transforming ssh messages to json and reverse ........................................... 16
Figure 7: Managing available systems .................................................................. 17
Figure 8: The widget in action, in the Congestion Control course page, during the UoP lab course ................................................................. 18
Figure 9: The RemoteGraphViewer widget in the web page of a course ....................... 19
Figure 10: The Initial Logins screen of the RemoteGraphViewer widget ....................... 20
Figure 11: The RemoteLogViewer widget in the web page of a course ....................... 21
Figure 12: The login screen of the RemoteLogViewer widget ...................................... 22
Figure 13: FORGE Architecture applied to iMinds’, with the widgets at the client side .... 24
Figure 14: Seamless widget integration (visualized here as ‘Exercise 4-1’) in a responsive course website ................................................................. 25
Figure 15: Overview of the widget architecture ...................................................... 26
Figure 16: All three components of the iMinds widget bundle in a single view (for Exercise 1-1) ................................................................. 27
Figure 17: Responsive view of the same widget bundle on a device with a smaller screen width 28
Figure 18: Example of an OMF Experiment Control widget with description and start/stop button 29
Figure 19: An advanced widget bundle with UI elements (here sliders) that change the experiment. ................................................................. 29
Figure 20: Example of an advanced description, with 3 sliders and linked data elements ...... 30
Figure 21: An example of a throughput graph based on the collected results of OML ........ 32
Figure 22: Jitter visualization of two IPerf streams captured with OML ......................... 33
Figure 23: The status widget showing the last 5 log entries of the OMF Experiment Controller 34
Figure 24: Plan Panel in LabWiki .................................................................. 36
Figure 25: Search previously saved document ...................................................... 36
Figure 26: Prepare Panel in LabWiki .................................................................. 37
Figure 27: Creation of a new experiment script in LabWiki. ..................................... 37
Figure 28: Editing Experiment Script in LabWiki ................................................... 38
Figure 29: Saving an OEDL script in LabWiki. ..................................................... 38
Figure 30: Execute Panel in LabWiki. ................................................................. 39
Figure 31: Start an experiment in LabWiki ............................................................ 39
Figure 32: Run of an experiment in LabWiki ....................................................... 40
Figure 33: LabWiki iBook widget creation ............................................................. 41
Figure 34: Resource disposition example

Figure 35: Wireshark software window

Figure 36: Wireshark widget prototype (under development)

Figure 37: Screenshot of the Cisco Packet Tracer for Windows.

Figure 38: Architecture of the PT server and client components and communication.
Executive Summary

This deliverable presents the initial set of educational widgets that have been developed by WP3 for the prototype lab courses of the project. These widgets make use of the FIRE testbed APIs that have been made available via the work done in WP2. The FORGE widgets are developed based on the learning contexts and requirements derived from the FORGE courses, as these are defined in WP5. Additionally, WP4 provides the methodologies and technologies used to develop and deploy the FORGE widgets via the FORGE infrastructure.

The FORGE widgets offer FIRE testbeds as educational facilities to learners and educators via a web interface. This means that the FORGE widgets can be accessed on the web via a Learning Management System (LMS) or any other web page. They can also be accessed via an interactive eBook, a first version of which has been released by FORGE and is available to download from the project’s web site\(^1\). The FORGE eBook features the FORGE widgets and other interactive and multimedia elements, such as quizzes and instructional videos, inside a collection of FORGE courses. A teaser video demonstrating some of these interactive elements is available via the project’s Vimeo channel\(^2\).

The FORGE widgets can be used via 3 different learning scenarios: they can be used in-classroom with the assistance of teaching assistants, or they can be used individually for remote learning, or as a set of self-assessment exercises succeeding traditional lectures.

It should be noted that this deliverable is primarily focused on the technological aspects associated with the development and deployment of the FORGE widgets. A more detailed presentation of the learning contexts and pedagogical aspects associated with the courses that make use of the FORGE widgets will be provided in D5.1 “Description and implementation of prototype lab courses” (M13).

The remainder of this report is structured as follows. First, an overview of the main technologies associated with the development of widgets for eBooks and Learning Management Systems is provided. Subsequently, we present in detail the FORGE widgets by introducing their learning contexts, the FIRE testbeds that they target, their implementation details, as well as the plans for their further development. Finally, the deliverable is concluded and the next steps of this work are discussed.

\(^1\) http://www.ict-forge.eu
\(^2\) http://vimeo.com/98431150
1 Overview of technologies

This section introduces the main technologies associated with the development of widgets for eBooks and Learning Management Systems. We also provide an overview of the most common and widely used standards related to these technologies. Additionally, we describe the approach followed by FORGE for the development and deployment of widgets.

1.1 Definitions

A **widget** is a micro-application performing a dedicated task. This task can be as simple as showing news headlines, but also more complex such as facilitating language learning or collaborative authoring.

Widgets can be either desktop-based or web-based. Desktop-based widgets reside locally on a computer and may access the Web for information, such as a desktop widget that shows the local temperature and weather. Web-based widgets reside on the Web and can be embedded on a web page, such as an RSS reader widget that fetches news on your start page. Web-based widgets have proven quite popular as they enhance the interactivity and personalisation of web sites.

The portability of widgets allows them to be embedded and used within different environments, either on the Web or the desktop. This has a great impact on the reusability of the learning solutions implemented as widgets.

Widgets, in general, consist of components built in XML, HTML, and JavaScript. XML is used to describe the specifications of the widget and contains instructions on how to process and render the widget. The XML description of the widget can contain all data and code of the widget, or it can have references (URLs) to the rest of the elements. HTML is used to deliver the static content of the widget. Finally, JavaScript is used in order to add dynamic elements to the widget, such as a message shown to the user as a response to an action he/she has performed.

A **widget bundle** is a set of widgets that complement each other and are utilised together for a common purpose. For example, a widget bundle for collaborative authoring can consist of widgets such as Google Docs and Google Talk.

A **widget store** is a repository of widgets. Widgets are commonly categorised within a widget store according to their purpose, e.g. widgets for planning, communication, and collaboration. Users can browse and download the widgets from the widget store, as well as provide feedback on the widgets in the form of ratings and comments.

An example of an educational widget store is the one developed by the EU FP7 project ROLE³. The ROLE widget store⁴ offers widgets for a variety of learning purposes, such as setting learning goals, planning and organising the learning process, finding suitable learning materials, collaborating with other learners, as well as reflecting on the learning process. All the offered widgets are categorised according to the learning task(s) that they target.

A **Learning Management System (LMS)** is an online software application offering facilities for student registration, enrolment into courses, delivery of learning materials to students, student assessment and progress monitoring. Popular examples of LMSs used by the academic as well as the business world include Blackboard⁵, Moodle⁶, and CLIX⁷.

---

³ http://www.role-project.eu  
⁴ http://www.role-widgetstore.eu  
⁵ http://www.blackboard.com  
⁶ http://moodle.org
One of the most widely used technical standards developed for ensuring the interoperability between the different LMSs is SCORM. SCORM stands for Sharable Content Object Reference Model. SCORM is primarily concerned with creating units of online learning materials that can be shared across LMSs. SCORM defines how to create Sharable Content Objects (SCOs) that can be reused in different systems and contexts. This means that, for example, a course exported in the SCORM format from one LMS can be imported and reused in a different LMS.

An electronic book or eBook is a book-length publication in electronic form, consisting of text, images, and (depending on the format used to publish it) videos and other interactive elements. eBooks can be read on computers or other electronic devices, such as tablets and smartphones or eBook reader devices.

One of the most common formats for publishing an eBook, supported by the majority of electronic devices, is ePUB. ePUB is a free and open eBook standard developed by the International Digital Publishing Forum (IDPF). ePUB is essentially an archive that contains the files comprising the eBook’s content, including HTML files, images, CSS style sheets and other assets, plus metadata that make the content available to be reliably consumed by any application or device compatible with the ePUB specification. In its latest version (3.0), ePUB (i.e. ePUB3) supports HTML5, allowing interactive elements like widgets to be embedded in the eBook.

On the other hand, Apple has developed a proprietary format for eBooks specifically for the iPad and MacOS 10.9 or later, called iBooks. The iBooks format allows the embedding of interactive elements inside an eBook, such as HTML5 widgets, thus enabling it to feature dynamic content offered by external online sources. The authoring software that we use for this purpose is the iBooks Author, an application for MacOS available to download for free. The application contains templates for galleries, quizzes, Keynote presentations, interactive images, and more. Alternatively, users can create and embed their own HTML5 widgets.

1.2 The FORGE infrastructure for widgets

As already presented in deliverable D4.1.1, widgets usually provide interactive web content to learners. This means that access should be provided by web containers and web services. The FORGE architecture defines a widget layer offered by FORGEBox. As shown in Figure 1, FORGEBox is the component that interconnects learning interactive content with FIRE resources. It offers a set of services that provide and host the interactive content of widgets, and interface with the FIRE resources via well-known FIRE APIs or with the Fed4FIRE portal. LMSs, eBooks and any future element that wishes to consume FIRE content, will need to discover reference points of widgets and Lab Courses descriptions. More details about the FORGEBox are available in D4.1.1 “FORGE architecture: Introducing FIRE to the learning community” (M6).

The FORGE widgets have been conceived as the means to present learning tools and infrastructure access in a more friendly way, via web tools independent of the underlying platform. The FIRE Adapters developed in WP2 wrap and adapt FIRE APIs to a set of common tools (scripts, libraries) used by these widgets. More details about the FIRE Adapters developed by

---

8 http://www.adlnet.gov/scorm
9 http://www.idpf.org
10 http://www.idpf.org/epub/30/spec/epub30-overview.html
11 http://www.idpf.org/epub/widgets
12 http://support.apple.com/kb/HT5068
13 http://www.apple.com/uk/ibooks-author

2014 © Copyright lies with the respective authors and their institutions.
FORGE will be provided in D2.1 “Adaptations made to the FIRE APIs to support education in prototype lab courses” (M11).

Figure 1: The FORGEBox birds-eye architectural view

The FORGE widget layer is not a specific service, but rather an abstract concept, since widgets can be provided by several web containers. For example, the Apache server or Tomcat can be considered as web containers. However, we expect that a local repository will keep track of the deployed widgets. FORGE course requirements define that widgets will be reused in an LMS environment or in eBook contents. The FORGEBox architecture does not enforce specific methods of implementing and deploying widgets. Widget developers may implement widgets in any programming language as long as they provide the best functionality and web access. The only constraint at this point will be to use open source tools and software. For example Apache and Tomcat could be used for web deployment and Java, PHP, ruby and Perl for widgets and their backend web service implementation.

The FORGE widgets are made available both for the web, so that they can be used inside LMSs or any other web page, as well as for eBooks. So far, the FORGE widgets have been featured in the web pages of the prototype courses offered by the project partners. FORGE will also offer a centralised web repository where all the project widgets and courses will be made available. Towards this goal, we are currently in the process of developing the FORGE Widget Store, called FORGESTore, an early alpha version of which is available online\(^\text{14}\). FORGESTore will host not only the FORGE widgets, but also the courses that use these widgets. Additionally, we plan to enable exporting the FORGE courses, including the widgets that these courses use, as SCORM packages, so that they can be easily imported and reused in other LMSs.

Regarding the availability of the FORGE widgets for eBooks, we have released an initial version of an eBook in the iBooks format, which contains a collection of the courses that are currently being developed by the project. In order to develop this eBook, we have made full use of the interactive

\(^{14}\) [http://www.forgestore.eu](http://www.forgestore.eu)
capabilities offered by the iBooks format, by developing and embedding in the courses included in the eBook a set of interactive self-assessment exercises, such as quizzes, as well as HTML5 widgets developed within FORGE that offer access to FIRE facilities (see Figure 2). As the FORGE courses and the associated widgets are further developed, this eBook is regularly updated. An ePUB version of this eBook for other devices is also expected to be available by M12.

Figure 2: A still from the FORGE iBook teaser video, demonstrating the use of an HTML5 widget inside the FORGE iBook
2 Prototype widgets

2.1 The ssh2web FORGECBox widget

One of the most used means to access machines residing in FIRE facilities is through terminal utilities. Planetlab\(^1\), VirtualWall\(^2\), BonFIRE \(^3\)cloud facilities, OFELIA \(^4\)Openflow testbeds, wireless access nodes and almost any other facility that offers full access to resources, offer experimentation via terminal access. The most known protocol is Secure Shell SSH\(^5\), since it enables an experimenter to access a resource and fully manage it via the command line. SSH client tools exist for almost any operating system, either natively in the system or via external downloaded tools (Figure 3).

As discussed already, having such a ssh client is the primary requirement for accessing most of FIRE resources. The next requirement is authenticating to the target system. FIRE facilities usually offer the two most common usages, via username and password combination or through public/private keys. The latter need some extra configuration from the experimenter for their clients, especially for generating the key pairs or reusing existing ones.

---

\(^1\) http://planet-lab.eu

\(^2\) http://ilabt.iminds.be/virtualwall

\(^3\) http://www.bonfire-project.eu

\(^4\) http://www.fp7-ofelia.eu

One of FORGE’s goals is to provide easy access to resources and reduce the difficulties of accessing FIRE resources. Moreover, FORGE tools and courses needed to be easily accessed through almost any device and means. Therefore, FORGE has selected the appropriate widgets for this process. Bringing ssh and terminal access to the web browser was a high requirement early in the project as arose from the first courses’ requirements. So we turned to web-based SSH technology. Web-based SSH makes it possible to access SSH servers through web clients that are based on JavaScript/AJax or JavaScript/WebSockets technologies. Moreover web-based ssh will solve some more issues like:

- Accessing SSH servers behind a firewall or proxy.
- IPv4, IPv6 connectivity issues.
- Learner’s session audit.
- Lab course creator user management access to resources.
- Key pair management and distribution to many resources.
- Auditability of all communication between the client and the SSH server which passes through the web application server and thus this communication can be logged. This prevents a malicious client from deleting logs of their activities. This also enables us to filter the commands sent by the client for security reasons.
- Web-based SSH implementations can be embedded into any web page allowing them to be integrated into other web-based applications, as well as other formats like ePUB.

As Figure 3 shows, in order to enable web-based SSH clients a server side web application is needed, hosted in a server playing the role of an ssh proxy. Incoming requests are processed on the web application server. Keyboard events are forwarded to a secure shell client communicating with the connected SSH server. Terminal output is either passed to the client where it is converted into HTML via JavaScript or it is translated into HTML by the server before it is transmitted to the client.

In the context of FORGE, the ssh proxy web application can be hosted in FORGEBox as a service. Our implementation is built upon another Java based project called Keybox\textsuperscript{20}. Within FORGE we made some refinements and enhancements to use for our needs, since this tool targeted users that want to manage cloud resources.

For example we program it to:

- Manage public key deployment.
- Redirect simple users to specific machines only.
- Create accounts in the background according to the rules of the username and later automatically assign them to machines (as it happens when developing courses in the case of Virtual wall).
- Enhance it with functionality to provide data for graphs widget and logging widget.
- Work also on simple http and ws protocols, to make it easier for our course scenarios.
- Filter the commands sent by the client for security reasons.

\textsuperscript{20} https://github.com/skavanagh/KeyBox
2.1.1 Learning context
The widget will be used in scenarios demanding console access to resources, in order to configure services, edit and execute scripts and programs, monitor log files, or run scripts for monitoring any kind of activity.

It will be used by the courses of “TCP Congestion Control” and “Advanced Network Architecture”. Mainly it will be the tool for allowing learners to modify system parameters, download utilities, edit scripts, and execute them. It will offer a hands-on experience on resources at the lowest level without any software installation. Figure 4 shows the widget displayed inside the web page of such a course.

2.1.2 Targeted FIRE testbed
The widget will be used by any FIRE testbed that provides resources accessible via terminal and command line tools. For the FORGE prototype courses of “TCP Congestion Control” and “Advanced Network Architecture”, the widget will be used by iMinds Virtual Wall and PlanetLab nodes. However, it is expected to be used by other testbeds offering wireless nodes.

2.1.3 Implementation
The widget is provided by a web application written in Java. Figure 5 displays the architecture of the widget’s web application.
Figure 5: Architecture of ssh2web widget

The java framework that is used to provide the interface is based on Apache Struts\textsuperscript{21}. The web container used is Jetty\textsuperscript{22}. Transport protocols for the web application to browser communication are http(s) and (secure) web sockets (ws/wss). For some current scenarios within FORGE we selected plain http and ws protocols, since encrypting the communication is not mandatory while it is very important to overcome some trusted signing issues on web browsers.

\textsuperscript{21} http://struts.apache.org/
\textsuperscript{22} http://www.eclipse.org/jetty/
Figure 6 shows how the communication is accomplished from the ssh2web application towards a remote machine, via private/public keys or even username/password pair. The text provided by the ssh communication is translated from a read/writer thread class into a json object and is passed to the websocket class, and is sent to the web browser via a JavaScript web socket opened on the browser. The reverse process is made when the user types a command. Each keystroke is passed as a json object to the websocket class, it is transformed again and sent through the ssh session class to the remote machine. Finally, we can audit the session for security purposes, for future reference, or any other kind of measurements by accessing the history of typed commands.

Deployment

As discussed already the Jetty web server is used as a container of our web app. Thus jetty needs to be present. To deploy it within FORGEBox, a simple copy of the target java application in Jetty will suffice. It is planned to ease the process by a quick installation as a service through the FORGESTore into the FORGEBox.

Usage

Access to the web application providing the widget depends on the user roles.

Learner access: In its current usage, the learner just needs to enter a username and password combination. Then he is immediately redirected to a specific machine, as given by the lab course designer or given URL machine name by his teacher. The learner can then type commands to the window, or drag and drop files from his PC over the widget in order to upload them to the remote machine.

Lab Course designer: The lab course designer can enter the backend web application interface, where there is an administration interface for a user account admin to configure the service in different aspects, such as systems, user accounts, profiles and any public ssh keys.
As an example, Figure 7 displays systems managed by our example installation in FORGEBox. These systems are connected to machines at the iMinds Virtual Wall. In much more complex scenarios the lab course designer needs to host the web application in his own FORGEBox installation and also configure the look and low level behaviour of the application by refactoring specific application's source files.

Usually since the widget will be used by including it into an iframe, the lab course designer just needs to include tags in the source code of the course’s HTML page, like, for example:

```
<iframe frameborder="0" width="500" height="350" scrolling="no"
src="http://www.forgebox.eu/ssh2web102" ></iframe>
```

Some extra parameters can be passed to the widget URL in order to configure some hidden fields. For example, if the following is added in the src of the iframe:

```
<iframe frameborder="0" width="500" height="350" scrolling="no"
src="http://www.forgebox.eu/ssh2web102/login.action?usertoconnect=patrasu
machineurl=labforge.wall2-ilabt-iminds-be.wall1.ilabt.iminds.be"
></iframe>
```

It will be possible to auto configure the widget to connect to a specific remote machine with a specific username.
Figure 8: The widget in action, in the Congestion Control course page, during the UoP lab course

Lab Course Assistant / Teacher: In some cases, the Lab Course assistant needs to configure the same entities as the Lab Course designer does. However, he is responsible for reserving or configuring the related FIRE resources of the course as well. Thus he needs to properly plan course execution and resources reservation. At this point specific FIRE adapters might be handy.

Prototype usage and testing

A prototype usage and testing of the SSH2WEB widget and web application, was performed during the first execution of a small part of the prototype lab course “TCP Congestion Control” at the University of Patras (Figure 8), on Tuesday the 27th of May and Monday the 2nd of June.

The goal of this lab course was to run a scenario with 150 nodes on VirtualWall. The service was used simultaneously by 50 users connecting via 3 instances of the widget. There were a total of around 150 simultaneous connections. The scenario was repeated 4 times, and thus the team was able to fix various issues. For example, some of the issues that came up related to many pool connections on the web application’s database or abnormal usage by typing wrong credentials and urls.

During this prototype course, an internal mechanism that autoregistered students according to their University ID was also created, to proper machines on the VirtualWall. For this scenario the Lab Course Assistant was previously reserved and created the 150 nodes on the VirtualWall.

2.1.4 Further development plans

The provided ssh2web web application providing the ssh2web widget will be further enhanced with some forms allowing Lab Course designers to restrict the commands sent to remote machines, thus to prevent some malicious usage by learners. Some provided metrics for each learner are also under consideration.
2.2 The RemoteGraphViewer FORGEBox widget
RemoteGraphViewer is a widget offering a line graph visualization of a file. It can also follow a generated file in real time while the values are generated. Figure 9 shows the graph within the “TCP Congestion Control” course. This widget is built on top of the functionality of ssh2web widget.

Figure 9: The RemoteGraphViewer widget in the web page of a course

2.2.1 Learning context
The widget can be used by courses that want to show some kind of a curve graph of measured values to learners. It is used by the “TCP Congestion Control” prototype course.

2.2.2 Targeted FIRE testbed
Currently we are targeting any machine that we have ssh access and can write to files. Thus there is no need for a specific FIRE testbed.

2.2.3 Implementation
The implementation is based on top of ssh2web. That is, the widget is provided by a web application that connects to a remote machine via ssh and transforms the contents of a file into a
json object. This json object is sent to the web browser via web sockets and can then be read by the graph component. The graph component is based on the implementations of Highcharts23.

**Deployment**

As discussed already the widget is based on ssh2web widget, thus follows the same process. It is currently deployed together with ssh2web, however a quick installation as a separate service through the FORGEStore into the FORGEBox is planned by the team.

**Usage**

Learners log in with a username and a password pair, like in the ssh2web widget. They also point to a registered remote machine, if necessary. They also need to type how to monitor the file and what the location of the file is (see Figure 10).

![Figure 10: The Initial Logins screen of the RemoteGraphViewer widget](image)

Since the widget will be used by including it into an iframe, the lab course designer just needs to include tags in the source code of the course’s HTML page, like for example:

```html
<iframe frameborder="0" width="100%" height="350" scrolling="no" src="http://www.forgebox.eu/ssh2web102/loginToGraph.action"></iframe>
```

**2.2.4 Further development plans**

It is planned to make it easier for the lab course designer to define the file to monitor and which values to plot. The login screen will also be refactored with easier functionality and hidden form fields so that the lab course designer can preconfigure the widget. Finally, a quick installation of the widget as a separate service through the FORGEStore into the FORGEBox is also planned.

---

23 http://www.highcharts.com
2.3 The RemoteLogViewer FORGEOBox widget

This widget visualises a file by performing a “tail” on a log file located on a remote machine and follows it in real time. Figure 11 shows a log from the “TCP Congestion Control” prototype course. The widget can display multiple log files simultaneously. This widget is built on top of the functionality of ssh2web widget and uses web sockets.

![Figure 11: The RemoteLogViewer widget in the web page of a course](image)

2.3.1 Learning context

The widget can be used by courses that want to show activity by log files to learners. It will be used by “TCP Congestion Control” prototype course, advanced network architectures course but also by wireless related courses.

2.3.2 Targeted FIRE testbed

As ssh2web and RemoteGraphViewer widgets, the RemoteLogViewer widget targets any machine that we have a connection over ssh. Thus, again, there is no need for a specific FIRE testbed, although it was tested on VirtualWall nodes during its implementation.
2.3.3 Implementation
The implementation is based on top of ssh2web widget. That is, the widget is provided by a web application that connects to a remote machine via ssh and then transforms the contents of a file into a json object. This json object that is equivalent to the monitored log file is sent to the web browser via web sockets and can then be read by the JavaScript based Logging component. The Logging component is implemented in JavaScript, using the jQuery library.

Deployment
It is currently deployed together with ssh2web, however it is planned to be offered as a separate service through the FORGEStore into the FORGEBox.

Usage
Learners log in with a username and a password pair, like they do in the ssh2web and RemoteGraphViewer widgets. They also point to a registered remote machine if necessary. They also need to type the absolute path location of the Logfile in the remote machine (see Figure 12).

2.3.4 Further development plans
It is expected to provide the availability of passing parameters to the src url in future developments, in order to let lab course designers configure the widget when designing the course. Offering the widget as a separate service through the FORGEStore into the FORGEBox is also planned. Additionally, we plan to decouple this widget from the ssh2web widget in order to gain more flexibility.

Figure 12: The login screen of the RemoteLogViewer widget

The widget will be mainly used by including it into an iframe, thus the lab course designer just needs to include tags in the source code of the course’s HTML page, like for example:

```html
<iframe frameborder="0" width="100%" height="350" scrolling="no" src="http://www.forgebox.eu/ssh2web102/loginToLogger.action"></iframe>
```
2.4 The WLAN widget bundle

This set of widgets enables learners to explore the characteristics of Wi-Fi communication (a.k.a. WLAN: Wireless Local Area Network) between wireless stations. The widgets control live experiments, using wireless nodes at a FIRE facility, for different scenarios of Wi-Fi communication. These scenarios involve one or multiple clients, various parameter settings etc. The results of the live experiments are displayed to the learner who can reflect about different theoretical concepts of Wi-Fi protocols to explain these results.

2.4.1 Learning context

The WLAN widget bundle is used in the iMinds prototype course “Achievable throughput on Wi-Fi networks”. These widgets are meant for undergraduate students studying for a Master degree in Computer Science Engineering. They can firstly be used in the course “Mobile & Broadband Access Networks” at the Faculty of Engineering and Architecture at Ghent University (Belgium), as these widgets are (1) a more user friendly enhancement and (2) an advanced port to FIRE facilities of practical lab sessions that were taught earlier within this course. Secondly, these practical sessions were also copied to the “Wireless networks and communications systems” course at the Electrical and Electronic Engineering School at Trinity College Dublin (Ireland) where these widgets can be used as well. These two use cases however are not exclusive. The widgets could be integrated in any other wireless networking course.

Regardless of the learning scenario that is used and the course they are used within, the purpose of these widgets is for students to better understand the impact on WLAN throughput of (1) the theoretical concepts and (2) the different configuration parameters. This is done by executing actual experiments on real hardware, which provide immediate feedback by displaying a live update of a graphical representation of the WLAN throughput that is achieved.

2.4.2 Targeted FIRE testbed

The iMinds prototype course “Achievable throughput on Wi-Fi networks”, using the WLAN widget bundle, is targeted at two testbeds of the iMinds’ iLab.t facilities24: the Virtual Wall and the w-iLab.t testbed. The Virtual Wall consists of 300 multi-core servers and the w-iLab.t testbed has amongst others 60 wireless nodes at fixed locations and 15 mobile nodes. When running the iMinds prototype course, one of the servers at the Virtual Wall is dynamically selected to serve as a web server hosting the widget bundle and interacts with three actual wireless experimentation nodes. These wireless nodes reside within the w-iLab.t testbed and are also dynamically configured.

While this course is optimized for the iMinds’ iLab.t facilities, the architecture and its implementation are developed in such a way that the course can (relatively) easily be ported to another FIRE facility with wireless nodes. The scripts and procedures that are facility dependent are separated from the general logic whenever possible.

2.4.3 Implementation

Figure 13 gives the entire architecture used in the student-experiment flow by the iMinds WLAN course (the course content will be described in detail in D5.1 “Description and implementation of prototype lab courses” (M13)). This architecture is in accordance with the general architecture as described in D4.1 “FORGE architecture: Introducing FIRE to the learning community” (M6) where specific scripting languages and tools are applied to, in order to suit the specific implementation needs of this course. The widgets consist of HTML5, CSS and JavaScript (top blue layer in Figure 13) and use PHP (orange layer in Figure 13) as a translation between the FIRE adapters and experiment execution logic running on the FORGEBox. In this deliverable, the widgets as well as their interaction with the PHP layer will be described. More information and implementation details

24 http://ilabt.iminds.be/
about the other layers (the FIRE adapters) can be found in D2.1 “Adaptations made to the FIRE APIs to support education in prototype lab courses” (M11).

![Diagram of FORGE Architecture applied to iMinds'] with the widgets at the client side](image)

**Figure 13: FORGE Architecture applied to iMinds’, with the widgets at the client side**

The iMinds widget bundle used for the WLAN course consists of three main components that can also operate as separate widgets: the **OMF experiment control widget**, the **OML result visualization widget** and the **OMF EC status widget**.

The main goal of the iMinds widgets is to allow a seamless integration of the FIRE experiment control and visualization on any platform they are used (a responsive course website, an eBook or an LMS). For this reason, the presentation of the widgets is based on the Bootstrap\(^{25}\) framework, giving a flexible HTML5 responsive basis for the visualization in many different environments. An example of this seamless integration is shown in Figure 14.

Figure 14: Seamless widget integration (visualized here as ‘Exercise 4-1’) in a responsive course website

The components working together to create these widgets are shown schematically below (Figure 15) in the overview diagram. The components in blue are responsible for the client side functionality, while the orange components are part of the server side FORGEBox.
As shown in Figure 15, *index.php* and *widget.php* are two different endpoints for the same *exercise.php* widgets. The main difference is that for *index.php* multiple exercise widgets (or bundles) are presented to the user, together with course material, figures and videos. In the case of *widget.php* only a specific exercise widget is loaded and the course material will be presented to the user from the calling platform (which is useful within e.g. an eBook or an LMS).

*Exercise.php* generates a single widget bundle, depending on the query parameter it receives from the calling function. The correct description is loaded and the widget is presented to the user. When the student performs an action on the widget, it will call the backend PHP scripts, depending on the widget, the exercise and the performed action. The flow of these actions is described in the next sections of this document.
Figure 16: All three components of the iMinds widget bundle in a single view (for Exercise 1-1)

Figure 16 shows the three components of the iMinds widget bundle, displayed on a full size desktop browser: the (1) ‘OMF experiment control widget’ comprises the commands to be executed and a Start/Stop button (upper left part in Figure 16), the (2) ‘OML result visualization widget’ contains a graph (upper right part in Figure 16) which is updated live during the experiment and the (3) ‘OMF EC status widget’ consists of blue information bars (bottom part of Figure 16) about the experiment’s progress. These different widgets are further explained in the next paragraphs. Depending on the viewpoint of the client the widgets will be reordered to maximize usability (i.e. responsive design). The same widget bundle is shown in Figure 17 on a smaller device (e.g. a smartphone), but all the separate components are equally legible.
OMF experiment control widget

This widget gives a student the control of an OMF experiment, with resources running on FIRE facilities (experiment handling is further explained in D2.1 “Adaptations made to the FIRE APIs to support education in prototype lab courses” (M11)). It consists of an HTML5 description and a start/stop button. The description component is explicitly broadly defined and without restrictions. It can contain simple text, figures, collapsible elements or even UI elements.

Collapsible elements are shown in Figure 18. They can be shown or hidden using a button that learners can click.

Figure 17: Responsive view of the same widget bundle on a device with a smaller screen width
Figure 18: Example of an OMF Experiment Control widget with description and start/stop button

Figure 19 shows an example of UI elements in the description. The advanced Exercise allows the student to adjust three sliders that correspond to three different values of a configuration parameter (in this case the packet size during WLAN transmission) that are used sequentially during the experiment.

Figure 19: An advanced widget bundle with UI elements (here sliders) that change the experiment.

Moving the slider will change the descriptive code in the description (see Figure 20). After linking the sliders to the experiment id programmatically (in the widget’s JavaScript) the value of the UI
element will furthermore be transmitted as experiment parameters to the FIRE adapters in the FORGEBBox.

![Experiment parameters](image.png)

**Figure 20: Example of an advanced description, with 3 sliders and linked data elements**

The start/stop button provides a simple experiment control to the student and its CSS class is dynamically adjusted by the widget’s JavaScript to reflect the experiment state. Therefore, the button can be in any of the following states:

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start!</td>
<td>disabled</td>
</tr>
<tr>
<td>Start!</td>
<td>ready</td>
</tr>
<tr>
<td>Stop!</td>
<td>experiment is running</td>
</tr>
<tr>
<td>✅ Start!</td>
<td>busy (starting or stopping)</td>
</tr>
</tbody>
</table>

Although the start/stop button is a very simple user interface, the logic behind it is by far the most complex of all widget components. Since this button is responsible for actually executing a real-life experiment from a shared web interface it is important that a user does not interfere with other
D3.1 LMS and eBooks widgets for prototype lab courses

users (please see D2.1 “Adaptations made to the FIRE APIs to support education in prototype lab courses” for more details). This is checked each time a user presses the start (or stop) button and an asynchronous call is made through experimentcontrol.js to the startexp.php (or stopexp.php) on the FORGEBox.

experimentcontrol.js contains all generic client side JavaScript, responsible for managing the course execution and visualization of the widgets. When other courses are based on these widgets they will reuse this collection of functions. These functions handle the (asynchronous) communication with the FIRE adapters and include:

- Starting an experiment
- Stopping an experiment
- Request experiment execution status
- Get and visualizing experiment results

startexp.php and stopexp.php are two examples of FIRE adapters, explained in detail in D2.1, which serve as an endpoint for the JavaScript on the client side to start or stop an experiment on a FIRE testbed through OMF.

OML result visualization widget

Once a student successfully starts an experiment, a continuous result polling mechanism starts at the same time that queries the FORGEBox on the results of the started experiment. The data is collected through a PHP script that is specifically tailored to the data structure of the collected OML data. This is a security-conscious design, where the client side has no direct access to database instructions, logic or code. For the current WLAN prototype course this leads to two implementations (see D2.1 for further details) for different exercises:

- getresults_iperf_transfer.php, returning an ordered array of time and throughput in Mbps
- getresults_iperf_jitter.php, returning an ordered array of time and jitter in ms
Figure 21: An example of a throughput graph based on the collected results of OML
Once data is received on the client side, it is visualized with the Highcharts\(^{26}\) JavaScript graphing library. This library was selected for its native HTML5 and responsive capabilities, but also for its flexibility and adaptability. This allows different graph types, tooltips and logic for each exercise if needed.

The process of the data collection itself is part of the OML backend, described in D2.1 “Adaptations made to the FIRE APIs to support education in prototype lab courses” (M11).

**OMF EC status widget**

Although this is the simplest widget in the bundle, it succeeds in giving the student a glimpse behind the scenes and strengthens the awareness that real hardware is used to perform the experiments.

---

\(^{26}\) http://www.highcharts.com
Once an experiment is started, the last 5 lines of the OMF Experiment Controller log file are continuously requested by the client, informing the user about the processes occurring behind the scenes, and eventually giving clues if an experiment would happen to fail. This is the exact same output that a user would see if he would perform the experiment himself on a FIRE facility, bridging the gap behind the ease of use of the experiment control and visualization widgets and actual experimentation on real hardware.

The OMF log info is returned in JSON format. If a user is not executing an experiment, he doesn't receive any status information, but a 403 HTTP error is returned.

2.4.4 Further development plans

The WLAN widget bundle is quite mature and it is stable to be used by learners. There is still room for some (optional) improvement, for example:

- Extend the number of WLAN exercises using this widget bundle
- Allow reservation of FIRE nodes via the widget bundle
- Reuse the widget bundle for other technologies than Wi-Fi
- Investigate the widget usage within the epub3 format

The reusability of this widget bundle will also be shown in new (prototype) course(s) which will exploit more advanced features of the FIRE facilities.
2.5 The LabWiki widget

This widget leverages the LabWiki tool to offer learners HTTP experiments in the current development while also allowing educators to setup their own LabWiki experiments and export them as widgets. LabWiki\(^{27}\) is a web-based interface developed by NICTA, which allows someone to design, describe and run an experiment using GENI\(^{28}\) resources. LabWiki is a web application, which provides the tools to plan, execute, observe and review series of experiments. It accompanies the entire lifecycle of an experiment-based study, recording each step along the way.

2.5.1 Learning context

This widget will be used in the context of wide networking area. In the particular case of the HTTP widget, the objective is to provide postgraduate students with the basic but essential knowledge on key concepts, currently used protocols, and emerging applications in networking systems. This widget can be used in any introductory course to network that will adopt a theoretical and practical approach to networking, with a strong emphasis on the latter. This course usually follows a top down approach to introduce the concept of networking layers and protocols. This will allow students with former networking training to refresh their knowledge, and students without such training to gain the essential understanding of networking systems.

We have used this lab in the context of a NICTA’s short course targeted at all new PhD students. This course usually ran over 2 days, with a total 8 hours per day, including lectures and laboratory work. The total contact time was thus 16 hours with a further 8 hours of self-directed laboratory work.

The course was divided in 4 sessions of directed learning of about equal size. The first session had a classic lecture format, and presents the key concepts and protocols. The remaining sessions combine both a short lecture part and larger laboratory part (i.e. lab). The first lab introduced the platform being used in this course and will propose experiments and questions focused on application-level protocols. The second lab focused on experiments and questions related to performances of transport-level protocols in a wireless environment, while the third lab focused on routing protocols in such environments.

The top-down approach adopted in this course was based on the classic “Computer Networking, A Top-Down Approach Featuring the Internet” textbook, by James F. Kurose and Keith W. Ross (Pearson, Addison Wesley). However due to the short lecture session, only a small subset of the textbook content was covered. Nevertheless, this lab can be used in any course that would follow this textbook.

2.5.2 Targeted FIRE testbed

This widget is based on the well-adopted tool LabWiki, which allows experimenter to prepare, run and analyze their experiment using Federated Resource Control Protocol (FRCP). This protocol has been developed in collaboration with FIRE and GENI consortia and is now considered as the standard protocol to orchestrate and run any kind of experiments in both FIRE and GENI. As such, the LabWiki widget is suitable for all testbeds in the FIRE.

In the particular case of the HTTP instance of this widget, the best suitable testbed would be testbeds such the iMinds Virtual Wall as it would be both cost effective and modular. Indeed, in this lab we introduce network impairments using emulation tools and therefore we do not need real physical machines.

\(^{27}\) http://groups.geni.net/geni/wiki/GEC18Agenda/LabWikiAndOEDL/Introduction

\(^{28}\) http://www.geni.net
2.5.3 Implementation

In this section, we present the two facets of the LabWiki widget. First, we present how the lab instructor can create a widget and integrate it inside an iBook. Then, we introduce the widget as seen by the student and detail its different components.

LabWiki for instructors

In order to use FIRE facilities in a course, teacher and lab instructors will first use the LabWiki tool to plan, prepare and test experiments to be used during lab sessions with students. Then once experiments are ready, they are able to export them in an iBook and ePub3 format. In this section we detail and illustrate the different steps that are necessary for the creation of these widgets.

Plan your lab. The Plan panel is where the instructor documents the lab and experiments. There they can describe for example experiment's notes, hypothesis, or models. In the context of FORGE, the lab instructor uses this panel to reproduce the lab description as found in Chapter 2 of the iBook to offer a web version of the lab. The plan panel is located on the left side of the LabWiki interface as displayed in Figure 24.

![Figure 24: Plan Panel in LabWiki](image)

To display a previously saved document, the instructor just types its name in the search field, for example this quickstart guide's name ‘quickstart.md’ as displayed in Figure 25.

![Figure 25: Search previously saved document](image)
Prepare your experiment. The Prepare panel allows the instructor to create, edit and view their experiment description, which is written using the OMF Experiment Description Language (OEDL)\(^\text{29}\). The Prepare panel is located in the centre of the LabWiki interface as displayed in Figure 26.

![Figure 26: Prepare Panel in LabWiki](image)

In order to create a new experiment description, the instructor needs to click on the "+" sign, enter a new filename, select "OEDL", and click on the "Create" button as displayed in Figure 27.

![Figure 27: Creation of a new experiment script in LabWiki.](image)

Similar to displaying previously saved lab description, in order to view and edit an experiment description, type its name in the search field, for example the simple experiment 'helloworld.rb' as shown in Figure 28.

\(^{29}\) http://omf.mytestbed.net/projects/omf6/wiki/OEDLOMF6
After editing their experiment, the instructor can save it by clicking on the disk icon in the prepare panel as illustrated in Figure 29.

Test your experiment. In the Execute panel, the instructor can configure and run your experiment, as well as view its progress and any defined graphs. The execute panel is located on the right side of the LabWiki interface as displayed in Figure 30.
In order to start an experiment, the instructor needs to load the experiment description in the Prepare panel. For example, they can load the `helloworld.rb` experiment, and then drag and drop the pen-and-paper icon from the Prepare to the Execute panel as shown in Figure 31.

The experimenter can now configure the parameters of their experiment, and press the Start button to run the experiment. While the experiment is running the instructor can view its properties, any defined graph, and any message it produces. These graphs will also be used later in the widget for the student to analyse the measurements. Figure 32 illustrates the example of the helloworld experiment.
Creation of the widget. Once the instructor has verified that the experiment works properly, he can now create the iBook widget. This action is performed by clicking in the execute panel on the wheel next to the search bar. In this panel the instructor fills the form as displayed in Figure 33. Once configured, the teacher is able to download the widget in a zip file and integrate it inside the iBook as described above.
LabWiki for students

The student sees the LabWiki widget in a similar way as other iBook widgets. Upon clicking on the widget, the client will connect to the LabWiki instance configured by the instructor as shown in Figure 20. Once connected, the students gain access to a simplified version of the LabWiki where they can only see the execute panel of the predefined script selected by the teacher. They can configure the experiment according to the lab description and click on the start button to run the experiment.

This configured experiment is then scheduled to be run as soon as resources are available. The students would be then able to access their results later by either going to the LabWiki instance or to another result widget currently under development.

2.5.4 Further development plans

As detailed above, the upcoming development plans comprise the implementation of a result widget, to enable offline access to the results of the experiments conducted through the LabWiki widget. As it stands, this widget can be used in any introductory course to network that will adopt a theoretical and practical approach to networking, with a strong emphasis on the latter. In particular this widget is well suited to any course following the classic “Computer Networking, A Top-Down Approach Featuring the Internet” textbook, by James F. Kurose and Keith W. Ross (Pearson, Addison Wesley).

We will also develop two additional widgets for the reliable protocols lab and for a post-graduate course. The reliable protocol lab will follow the transport layer chapter from the aforementioned textbook. In the postgraduate course, students will learn techniques for performance evaluation of distributed systems. These techniques will then be applied to designing systems to have good performance, and to the analysis of future workloads and the system changes required to cope with them.

We will also integrate the LabWiki widget within the FORGEBox and we will extend the widget creation for ePub3.
2.6 The PlanetLab widget
This widget will provide access to the PlanetLab\textsuperscript{30} testbed via a graphical interface.

2.6.1 Learning context
This widget will be used in the UPMC network courses focused on the impact of geographic distance in HTTP traffic. In these courses, the student will experience how the localization of services can impact digital communication.

In this context, students will have access to a resource used as a client and several resources used as a web server. They will then generate HTTP traffic using those resources and dissect this traffic.

![Figure 34: Resource disposition example](https://www.planet-lab.eu)

This widget will provide an experiment controller for the course, allowing students to gather network traces and then analyse them. This widget will be initially used for in-class learning, but can be then extended to online learning too.

2.6.2 Targeted FIRE testbed
This widget will be used to control PlanetLab nodes. It will interact with these nodes through an ssh connection.

\textsuperscript{30} https://www.planet-lab.eu
2.6.3 Implementation
This widget will provide a web interface based on php and javascript. These web parts will interact with a bash script in charge of running the experiment. The widget will provide detailed status information and then send a pcap file that can be further analysed with the Wireshark widget, which is described in the following section.

2.6.4 Further development plans
This widget is still under development and will be used to provide additional course use cases, which will be presented in D5.1 “Description and implementation of prototype lab courses” (M13). We will also work on the integration of the PlanetLab widget into the FORGEStore environment.
2.7 The Wireshark widget
This widget will provide a web graphical interface to visualize network traces.

2.7.1 Learning context
This widget will be used in the UPMC prototype course on the impact of geographic distance on HTTP traffic. It will allow students to dissect the packets generated on a testbed. This widget will provide an interface based on the well-known software Wireshark.

Figure 35: Wireshark software window

http://www.wireshark.org/
This software provides a graphical interface where the student can visualize the different protocols involved in network transmission. It provides three main panels, as illustrated in Figure 35:

- The first (top) panel lists the packet with some details.
- The second (middle) panel provides the list of protocols involved in this packet and details several fields.
- The third (bottom) panel is used to show a hexadecimal version of the packet.

### 2.7.2 Targeted FIRE testbed

This widget will be initially used in a course designed for PlanetLab. In the future, this widget could also be used in various courses involving other FIRE testbeds.

### 2.7.3 Implementation

This widget is developed in JavaScript, using some additional libraries such as Jquery\(^\text{32}\) and Datatables\(^\text{33}\). The idea is to keep it mainly pure JavaScript in order to allow compatibility with most modern browsers.

The packet description file is based on libpcap\(^\text{34}\) which is a library used for network traces capture. This library is used by several software applications and provides a common way to save network traces into file.

\(^{32}\) [http://jquery.com/](http://jquery.com/)

\(^{33}\) [http://www.datatables.net/](http://www.datatables.net/)
2.7.4 Further development plans

The initial plan is to get a full support of the HTTP protocol so the student could use it in the UPMC courses. The widget will handle IPv4 and IPv6 connectivity, as well as the TCP stream.

In the future, the widget can be extended to support other protocols. This will allow the widget to be used in other courses involving different network protocols. This widget could also be used in various courses involving other FIRE testbeds.

We will also work on the integration of the Wireshark widget into the FORGEStore environment.

34 http://www.tcpdump.org/
2.8 The Packet Tracer widget

The Packet Tracer widget will enable learners to simulate different types of networks and understand how network technologies work and interact with each other. Packet Tracer (PT) is a Windows/Linux network simulator developed by Cisco (see Figure 37). It is currently being adapted as a widget by the Open University in collaboration with Cisco.

![Figure 37: Screenshot of the Cisco Packet Tracer for Windows.](image)

2.8.1 Learning context

The PT widget is being developed for use in the “Basic Network Router Configuration” prototype course. This course introduces learners to the basics of simulated networks using PT from the Cisco Systems Networking Academy. Within this course, the PT widget will be used by learners in order to simulate different types of networks and understand how network technologies work and interact with each other.

The target audience of this course is learners who want to gain the required qualifications towards becoming Networking Engineers. The purpose of this course is to teach the basics of simulated networks and the use of Cisco network equipment. In particular, learners who study this course will acquire an understanding of the following:

- The basics of IP addressing.
- How the role of a router and a switch can be established.
- What the server technology is
- How to configure IP addressing on a personal computer.
- How to send traffic from one device to another across a network.
- How to configure a router, with an address and a hostname.
- How to use router ‘show’ commands to check its status.
- How to configure a router via its command-line interface.

---

2.8.2 Targeted FIRE testbed
This course does not target a particular FIRE testbed, but aims to connect the FIRE community with the Cisco Networking Academy. The Cisco Networking Academy\(^\text{36}\) delivers a curriculum to help students develop the foundational ICT skills needed to design, build and manage networks. In order to deliver this curriculum, Cisco partners with educational institutions, non-profits, nongovernmental organizations, as well as community centres worldwide that provide classroom space, computer lab equipment, and qualified instructors. Cisco provides online curricula, virtual learning tools, instructional support, teacher training, and professional development opportunities for instructors. The Cisco Networking Academy has more than 1 million students worldwide, taught by 9,000 participating academics from 170 countries.

The Open University (OU) is one of the partnering institutes of the Cisco Networking Academy in the UK, delivering a variety of networking courses to OU students. Building upon this partnership in the context of the FORGE project, we aim at exposing the millions of students of the Networking Academy to the educational services offered by the FIRE testbeds. This will be achieved by presenting the FIRE testbeds and the FORGE prototype courses as the follow-up from using PT. In this way, students of the Networking Academy will use the FIRE testbeds in order to apply the simulation results acquired by PT on real-life networks.

2.8.3 Implementation
We are currently in the process of developing a widget to embed a live connection to a running instance of PT. This section outlines the requirements, design and current status of this widget.

**Requirements**
Because the main audience for iBooks is on Apple’s iOS, there are restrictions that must be met. In particular, the only available languages are HTML and Javascript, and so all interactive elements must be expressible in these. At least up to the current version, iOS 7, there is no way of embedding the interface of one iOS app inside another\(^\text{37}\), and so it is not possible to write any interactive code in languages such as Objective C and deploy it in an app to be installed alongside the iBook.

These restrictions impose the requirement that the PT widget for the iBook must be implemented purely in HTML and Javascript. This requirement also complies with the additional requirement of making the PT widget available as a web widget for use inside an LMS. By developing the widget in HTML and Javascript, it can be easily embedded in an LMS or any other web page.

**Design**
In order to meet the above requirements, the widget is implemented primarily with a server-side backend, with which a client user interface can communicate using HTTP. Figure 38 illustrates the architecture of the whole system.

Full PT support will be developed incrementally, beginning with simply displaying the contents of PT instance populated with a basic set of devices and progressing to more sophisticated interactions.

**Implementation**
Server-side: The server that handles communication with the client is written in PHP. Requests from the user are submitted over HTTP with results returned in plain text.

---

\(^{36}\) [https://www.netacad.com](https://www.netacad.com)

\(^{37}\) The upcoming iOS 8 may relax this limitation for apps, but at the time of writing, it is not clear whether iBooks will be affected.
The PHP component communicates via a text protocol with a standalone “bridge” server written in C++. The bridge server uses Cisco’s IPCFwCPP library, which is designed to expose an IPC (Inter Process Communication) interface to a running instance of PT, and translates the client’s requests to PT and vice versa.

**Client-side:** As discussed above, the client is written in HTML and Javascript, and consists solely of the user interface code. The client connects to the PHP server-side code to retrieve the current state of the currently running instance of PT, and displays it to the user, to whom it will simply appear as an interactive widget. Any actions the user performs on the displayed network topology will be communicated to the server, and the results displayed.

**Figure 38: Architecture of the PT server and client components and communication.**

### 2.8.4 Further development plans

The basic framework for both client and server sides is implemented, with a very basic interface in HTML capable of loading the current status of a PT instance via the server and displaying it to the user. Currently the client side has only been tested in a desktop web browser with the iBook embedded widget expected to be working soon.

By M12, we anticipate being able to demonstrate significant interaction with PT via the client running within the iBook and as a web widget available via FORGEBox.
3 Conclusions and next steps

This deliverable presented the set of educational widgets that have been developed so far by the FORGE partners in the context of WP3. This deliverable was focused on the technological aspects associated with the development and deployment of the FORGE widgets. In particular, we presented in detail the FORGE widgets by introducing their learning contexts, the FIRE testbeds that they target, their implementation details, as well as the plans for their further development.

The FORGE widgets expose FIRE educational facilities to learners and educators via a web interface, which can be accessed on the web (e.g. via a Learning Management System) or alternatively via an interactive eBook. We have released an initial version of an interactive eBook in the iBooks format for the iPad and MacOS, featuring the FORGE widgets and other interactive and multimedia elements, such as quizzes and instructional videos, inside a collection of FORGE courses.

Regarding the next steps of this work, we are currently in the process of developing a centralised web repository, FORGEstore, where all the project widgets and courses will be made publicly available to learners and educators. We are also working towards targeting additional educational delivery channels by releasing the FORGE eBook in the ePUB format.

We are continuously refining the specifications of the FORGE widgets based on the updated educational requirements of the prototype courses that are currently being developed by WP5. Based on the refined specifications, we will continue updating the implemented widgets and develop additional ones in order to address new educational requirements.
APPENDICES

Appendix A – Acronyms list

API – Application Programming Interface
CSS – Cascading Style Sheets
FIRE – Future Internet Research and Experimentation
HTML – HyperText Markup Language
HTTP – Hypertext Transfer Protocol
IP – Internet Protocol
JSON – JavaScript Object Notation
LMS – Learning Management System
OEDL – OMF Experiment Description Language
OMF – cOntrol and Managment Framework
OML – OMF Measurement Library
SCORM – Sharable Content Object Reference Model
TCP – Transmission Control Protocol
UI – User Interface
WLAN – Wireless Local-Area Network
XML – Extensible Markup Language