Facilitating student understanding of Internetworking via e-learning

ANDREAS KOKKALIS

Master's Thesis at ICT
Supervisor: Anders Västberg
Examiner: Gerald Q. Maguire Jr.

TRITA xxx yyyy-nn

Abstract

Learning Management Systems (LMSs) are widely used in higher education to improve the learning, teaching, and administrative tasks for both students and instructors. Such systems enrich the educational experience by integrating a wide range of services, such as on-demand course material and training, thus empowering students to achieve their learning outcomes at their own pace.

Courses in various sub-fields of Computer Science that seek to provide rich electronic learning (e-learning) experience depend on exercise material being offered in the forms of quizzes, programming exercises, laboratories, simulations, etc. Providing hands on experience in courses such as Internetworking could be facilitated by providing laboratory exercises based on virtual machine environments where the student studies the performance of different internet protocols under different conditions (such as different throughput bounds, error rates, and patterns of changes in these conditions). Unfortunately, the integration of such exercises and their tailored virtual environments is not yet very popular in LMSs.

This thesis project investigates the generation of on-demand virtual exercise environments using cloud infrastructures and integration with an LMS to provide a rich e-learning in an Internetworking course.

Strict hyphenation breaks the template, fix this.

Feedback: make abstract 1 column wider

Sammanfattning

Add swedish section

Acknowledgements

I would like to acknowledge ...

Contents

List of Figures

List of Algorithms

List of Tables

Listings

List of Acronyms and Abbreviations

Keep alphabetical order!

AJAX Asynchronous JavaScript and XML

AWS Amazon Web Services

CA Certificate Authority

CS Computer Science

DSL Domain Specific Language

e-learning electronic learning

EC2 Elastic Compute Cloud

ERB Embedded RuBy

GLUE! Group Learning Uniform Environment

GUI Graphical User Interface

HTML Hyper Text Markup Language

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IT information technology

JSON JavaScript Object Notation

KTH Kungliga Tekniska Högskolan

LIS Learning Information Services

LMS Learning Management System

LTI Learning Tools Interoperability

LXC Linux Containers

MIME Multipurpose Internet Mail Extensions

MOOC Massive Open Online Course

OCI Open Container Initiative

SCROM Sharable Content Object Reference Model

SHA Secure Hash Algorithm

SSH Secure Shell

TC Tool Consumer

TLS Transport Layer Security

TP Tool Provider

URL Uniform Resource Locator

XML Extensible Markup Language

Chapter 1

Introduction

The use of electronic learning (e-learning) technologies has been well established in modern education to assist both students and instructors in their learning, teaching, and administrative tasks. One of the e-learning technologies most widely adopted by the academic community is Learning Management Systems (LMSs). A LMS is a software application that handles all aspects of the learning process [1], enabling instructors to design rich e-learning courses and students to experience self-paced learning using a variety of features, such as on-demand course material, video lectures, automatic delivery and evaluation of assignments, collaboration tools, etc.

Many courses, especially in various sub-fields of Computer Science depend on training events in the form of programming assignments, laboratory exercises, simulations, etc. These activities are crucial for students to gain hands-on experience with complex concepts and systems [2]. Although LMSs support on-line training events, such as interactive quizzes with automatic evaluation and analysis of results, providing training events that depend on using complex virtual environments and software are not yet very popular (and hence not widely supported or used).

One of the main advantages of using an LMS is that it supports the integration of external applications to provide personalized, domain specific e-learning, such as messaging and video streaming services, on-line office suites, collaboration tools, or even training environments with exercises tailored to the needs of a specific course.

1.1 Background

Hands-on experience is very important to achieve understanding of complex systems and concepts. For example, when studying computer networks, laboratory exercises are a common student activity. An Internetworking course often involves students studying the performance of different Internet protocols under different conditions (such as varying throughput bounds, error rates, and patterns of changes in network conditions).

These experiments depend on specific software, network topologies, and local or virtual hardware. Traditional approaches for realizing such environments depend upon the student's own hardware or on-site computer labs with pre-configured software[3]. More modern approaches involve remote access to virtual machines running on central servers or cloud infrastructures [4].

Currently LMSs do not have built-in support for such laboratory environments. However, one of the main advantages of designing an on-line course on top of an LMS that supports the integration of extenal applications is to provide tailored functionality for the course's and student's specific needs. Today, many LMSs, such as Instructure Inc.'s Canvas [5] LMS, implement the IMS Global Learning Consortium Tools Interoperability® (LTI®) specification. Learning Tools Interoperability (LTI) allows the exchange of information between the LMS and third party components, thus exposing internal functionality of the LMS to external applications in a controlled manner.

Supporting virtual laboratory environments in a LMS in order to meet the needs of an Internetworking course, requires the design of a software framework that implements the LTI interoperability specification in order to exchange relevant information between the laboratory environment and the LMS.

1.2 Problem definition

Hands on experience is very important aspect of the learning process in several fields of Computer Science, including computer networks. Understanding the domain specific concepts and problems of an Internetworking course, depends greatly on exercise material and laboratory practice. Today, such exercises, are not usually designed to extract suitable analytics for the instructor (as an instructor ideally wishes to evaluate each student's level of understanding of each of the different concepts covered in an exercise). Assessing the student's understanding is currently achieved ~~via~~ using additional training material, such as quizzes or assignments in ~~forms~~ of reports which are manually evaluated by instructors or by other students in the form of peer review~~ing~~. These alternative methods both introduce a delay in feedback to the student (hence reducing the student's rate of learning) and are not scalable (for example, preventing their use in Massive Open Online Courses (MOOCs)).

Supporting an on-line version of an Internetworking course through a LMS that enables students to achieve the course's learning outcomes at their own pace, depends greatly on designing interactive practice environments. Such environments should be easily modified by the instructor to fit the needs of different exercises. Moreover, the environment must provide feedback for both students and teachers. Although today LMSs support a variety of training events, such as quizzes and assignments through integration of external services, on-line virtual laboratory environments that fulfill the requirements of an Internetworking course are not yet well supported and hence not widely used.

However, similar practice environments are common in on-line courses that teach programming languages. Such environments are part of systems that provide tools for designing coding assignments, and support several assessment methods, including automatic evaluation and grading of code[6] and programming quizzes. These systems often provide standalone web applications or LTI integrations in LMSs that expose functionality for developing code, submitting assignments, and presenting feedback to users[7, 8].

This project aims to design a software framework that supports interactive training material for an Internetworking course that extracts suitable analytics of the learning process for both students and instructors, and integrates with a LMS to provide a rich e-learning experience.

1.3 Goals

The design of such a laboratory environment for an Internetworking course has to meet several user requirements from the perspective of both students and instructors, and integrate with a LMS to offer a rich e-learning experience. The expected outcome of this project is a software framework that supports instantiation of on-demand laboratory environments using cloud based technologies to enrich the learning experience of students, allowing them to proceed at their own pace. Additionally, the framework should enable a teacher to customize the environment according to different exercises' requirements, and provide the instructor with constructive feedback about each student's progress and understanding.

The process of designing this framework can be realized by achieving the following goals:

- Devise a method to easily build virtual laboratory environments,
- The framework should support evaluation and analysis methods to be applied to the exercise in a way that is useful for both instructors and students.
- The framework should be integrated with the LMS to enable students to access the training environments via the LMS,
- The method of integration of such exercise environments should be usable by others - thus an important part of this thesis project is documenting the selected method to facilitate the integration of a diverse set of external environments (for example, an ns-3 simulator configured for a particular simulation),
- The framework should scale in such way that enables students to do assignments at any given time, thus offering on-demand availability of the underlying services, and
- A student should be able to access a training environment within an upper bounded time from the moment he requests to start an assignment ~~from the LMS~~.

1.4 Research Methodology

Design science research addresses important unsolved problems in unique or innovative ways or solved problems in more effective or efficient ways. It focuses on the design and construction of information technology (IT) artifacts that have utility in real-world, application environments. The artifacts, as the outcome of the research process, aim to improve domain-specific systems and processes [9, 10]. The utility, quality, and adequacy of a design artifact, is thoroughly evaluated under varying experimental setups to verify that it successfully fulfills the requirements.

Design, in several research fields, including IT, is an iterative process of planning, generating alternatives, and selecting a satisfactory outcome. Design science research, although it is not performed using strictly defined processes, can be summarized by three closely related cycles of activities (these cycles are the relevance cycle, the rigor cycle, and the design cycle)[11], that act as guidelines for designing, constructing, and evaluating an artifact. The relevance cycle establishes the application context that not only provides the requirements for the research as inputs, but also defines acceptance criteria for the evaluation of the research results. The rigor cycle provides past knowledge to the research project to ensure its innovation. It is contingent on researchers to thoroughly research and reference this knowledge base in order to guarantee that the designs produced are research contributions and not routine designs based upon the application of well-known processes. The central Design Cycle iterates between the core activities of building and evaluating the design artifacts and processes of the research [9], until the acceptance criteria, as defined in the Relevance Cycle, are met.

This project is carried out using the design science research approach. The resulting software and documentation attempt to solve the problem of designing and realizing a framework for rich on-line laboratory environments for an Internetworking~~,~~ ~~e-learning course~~ that is accessible via a specific learning management system (Canvas LMS). The two different domains that define the context of this problem are the Internetworking course domain, and the LMS along with the method(s) of integration of external applications into Canvas (in this case via LTI).

1.5 Deliminations

This project addresses the problem of designing and integrating virtual laboratory environments to support e-learning in an LMS for an Internetworking course. The laboratory framework, the expected outcome of this project, has to fulfill several requirements: usability for different types of users (instructor, administrator, and student~~,~~), integration into the Canvas LMS via the LTI specification, and satisfy the laboratory and pedagogical challenges of this particular course. Although there are different specifications for integrating external applications and services into a LMS [12], this project addresses only the LTI specifications, as this method is supported by Canvas (along with many other LMSs, for example LTI can be used together with edX as either a consumer or producer[13]). The design of the laboratory framework, is designed to suit the needs of a typical classroom (in this case approximately 30 students), thus its scalability is limited.

Testing the scalability of the designed system regarding the number of users is outside of the scope of this thesis project. However, a system might be scaled up on Amazon Web Services (AWS) by using larger instances (vertical scaling) or by creating multiple instances (horizontal scaling). Additionally, scaling up and down of services in clouds has been invested by others [14].

TODO: Internetworking assignments, and extraction of relevant analytics have also limitations. Make sure that they are reflected in this section.

1.6 Structure of the thesis

Chapter 2

Background

This chapter explains what is a LMS and how learning applications are integrated in such systems to support rich e-learning. Moreover, it introduces research artifacts that offer on-line training environments for various courses in Computer Science (CS) domain. Lastly, it introduces ~~the~~ technologies that were used to design the framework that supports training events for an Internetworking course.

Section 2.1 ...
Finilize this paragraph when this chapter is complete.

2.1 LMS

LMSs are software applications that automate the training, teaching, and administrative tasks of the learning process [1]. They have been widely adopted by higher education institutions to automate their organizational functions and provide a rich e-learning experience for both instructors and students.

Such systems are designed to provide self-guided services; rapid delivery and composition of learning material; tracking and reporting of progress through training programs, classroom, or on-line events; personalized content; and centralization and automation of administration [15]. From a learner's perspective the most common use cases of a LMS are planning ones own learning experience and collaboration with colleagues; while from an instructor's perspective the most common use cases are the design and delivery of educational content along with tracking and analysis of students' learning evolution [16].

The main functionality of a LMS concerns content organization and delivery, communication and collaboration, and assessment* of student's learning process. Some of the most commonly used features of an LMS for e-learning are video stream-

*Formative assessment is performed by teachers during the learning process, to modify and improve the teaching and learning activities. It is based on observation of students' individual efforts and development; thus, having a qualitative and diagnostic nature. Summative assessment, performed by both instructors and students, is based on public criteria that aim to measure student's achieving of the course learning outcomes. ~~[17]~~

ing of lectures, on-line notes and presentations, quizzes and practice environments, automatic evaluation of assignments (usually exercises with predefined input and output), wikis, discussion forums~~, . . .~~ [8]. These services are either offered directly by the LMS or by integrating external applications that are designed according to specific interoperability standards. Section 2.2 describes this interoperability and integration in detail.

Although LMSs provide built-in learning applications for designing e-learning courses, their functionality is often very limited and might not suit the needs of every course. Moreover, not all LMSs support the same learning tools, nor provide the same functionality for e-learning. On the contrary, external learning tools can be integrated with multiple different LMSs, and allow re-using existing material thus minimizing the effort for designing an e-learning course. Usually such tools are web services[†] that are discoverable by an LMS via the service Uniform Resource Locator (URL) and authorization parameters such as secret keys. The communication between the LMS and the tool is performed by exchanging messages whose format and content is defined by the interoperability specification. Section 2.3 shows several web frameworks can be used to design external learning tools as web services.

There are several LMSs in the market (Blackboard, Moodle, Kanu, . . .) that are used by multiple institutions. In the scope of this project the chosen learning management platform is Canvas [5]. This LMS was chosen because the system is open source, supports a well defined interoperability specification, and was selected ~~earlier this year~~ by KTH as their ~~future~~ LMS.

2.2 LTI

Interoperability is the ability to communicate, execute programs, or transfer data among functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units [19]. An e-learning platform usually consists of several services such as course and user administration modules, and learning applications that exchange information in a formal and standardized way.

The IMS Global Learning Consortium Tools Interoperability (LTI) specification establishes a way of integrating rich learning applications (often remotely hosted and provided through third-party services) with platforms, such as LMSs, portals, learning object repositories, or other educational environments [20]. The main goal of LTI is to standardize the process of building links for sharing information and exposing functionality between external learning tools and the LMS [21]. There are two major pieces of software involved in LTI. The first is called Tool Consumer (TC)

[†] In service oriented architectures, a web service is a piece of software that makes itself available over the Internet and allows third-party software to communicate with them by exchanging strictly defined messages formatted in Extensible Markup Language (XML), JavaScript Object Notation (JSON), etc.

and it refers to the software (such as an LMS) that consumes the output of external tools, and the second, is ~~the~~ Tool Provider (TP) which provides an external tool for use by the TC.

An example of a basic learning tool, is a service that accepts a request to perform a course assignment such as multiple choice question via a web form, evaluates the user input, and returns a pass/fail grade. In this scenario, the service is the TP and Canvas LMS is the TC. A user of Canvas with administrative access (e.g teacher), ~~is required to~~ configure the integration of the external tool, a course assignment for which the tool will be launched, and finally, ~~choose~~ whether the interface of the tool will be embedded in Canvas~~,~~ or run in a new browser window. Figure 2.1 shows a basic flow for launching a TP from the TC. The user requests the LMS that they want to do an assignment. This specific assignment has been configured to launch a specific LTI capable external tool together with arguments that are passed to the TP. The TP authenticates and accepts the LTI Launch request by the TC and starts a session for that particular user that allows this user to interact with the assignment.

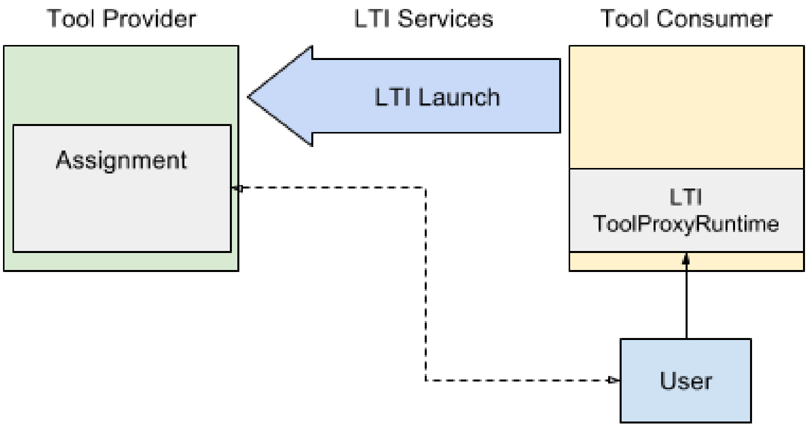

Figure 2.1: User launching an external tool

A TP often requires access to course related information, such as people, groups, memberships, courses, and outcomes. This information along with standardized ways of retrieving it are defined by the IMS Global Learning Consortium Learning Information Services (LIS) specification [22]. These services can be provided either by the TC or by a third party system. Canvas LMS implements the LTI version 1.1 which includes a subset of the LIS specification, called the LTI Basic Outcomes Service. In the example mentioned above, the information that Canvas provides to the TP when performing an LTI Launch are: how to access the LIS services, the resource identifier (assignment) for which a grade will be reported, and user information such as the unique identifier of the student. Figure 2.2 shows how a TP can communicate with LIS services to get user data and report the grade of the assignment back to the TC.

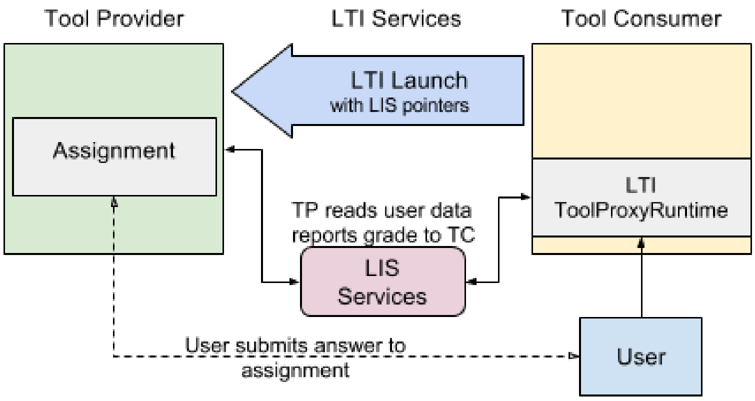

Figure 2.2: A TP using LIS services

2.3 Sinatra DSL

A simple web server is a piece of software designed to process Hypertext Transfer Protocol (HTTP) requests. Many web frameworks have been developed in several programming languages that allow to quickly develop web servers and applications. Sinatra[23, 24] amongst them, is a Domain Specific Language (DSL) for writing web applications in Ruby. A Sinatra web application is organized around routes which are HTTP methods paired with a URL-matching pattern. Listing 2.1 presents a minimal sinatra application. The route `"/"` is paired with a `get` HTTP method. Every time this route is invoked, it provides a `"Hello World!"` text response.

```
# hello_world.rb
require 'sinatra'

get '/' do
  'Hello world!'
end
```

Listing 2.1: Sinatra basic route

A file named `hello_world.rb` contains the code shown in Listing 2.1, which is called a route block. A route block starts with a keyword such as `get, post, put, ...` and corresponds to an HTTP method, and finishes with the keyword `end`. Executing the web application is as simple as running the command `ruby hello_world.rb`. This will start a sinatra web server on the default host (`localhost`) that listens for ~~tcp~~ connections on the default port (`4567`). By visiting the ~~url~~ `http://localhost:4567/` ~~on~~ a browser, the route `"/"` is invoked and the response returned to the user.

A route can also utilize HTTP GET query parameters as shown in Listing 2.2. In this case, if a `course_id` is provided as a parameter of query string, then its value is loaded into the local variable `courseID`. The same concept could be applied if the route was a ~~post~~ HTTP method and course_id was one of the ~~post~~ parameters.

```
get '/assignments' do
  # matches "GET /assignments?course_id=IKXXX"
  courseID = params['course_id']
  # uses course_id variable; query is optional to the / route
end
```

Listing 2.2: Sinatra route with HTTP GET parameters

Sinatra also supports the use of wildcards to match all parameters of the query string. Such parameters are called splat, are symbolized with a "*" router pattern, and are accessible via the `params['splat']` array. In the Listing 2.3, the route `'/department/*/course/*'` represents the course catalog of a university. The splat parameters match the department (`informatics`) and course (`ID001`) identifiers respectively.

```
get '/department/*/course/*' do
  # matches /department/informatics/course/ID001
  params['splat'] # => ["informatics", "ID001"]
end
```

Listing 2.3: Wildcard route pattern

Templates are a text injection mechanism, that allows static text to be enriched using dynamic content (e.g an Hyper Text Markup Language (HTML) template might contain some static text and variables, where the variables are replaced during runtime). In Sinatra a template by default is stored under the directory `views`, and can be used in many different ways, including rendering HTML pages, constructing a JSON object as a response to an HTTP request, etc. Listing 2.4 shows the route `get '/assignments'` which stores the value of the `course_id` parameter into an instance variable `@courseID` which makes the value of this variable available for use in the template shown in Listing 2.5.

```
get '/assignments' do
  @courseID = params['course_id']
  erb :index
end
```

Listing 2.4: Sinatra route with template

Calling the `assignments` route by visiting the url `http://localhost:4567/assignments?course_id=IK1552` will parse the query parameter, invoke the `index.erb`[*] template stored under the directory views, and substitute the text `<%= @courseID%>` with the value of the variable `@courseID`.

[*] Embedded RuBy (ERB) is part of the Ruby standard library, and serves as the mechanism for variable substitution within template files.

The response that will be rendered by the browser will be an HTML page that contains the text "List of assignments for IK1552" in its body.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Assignments</title>
  </head>
  <body>
    <p>List of assignments for <%= @courseID%></p>
  </body>
</html>
```

Listing 2.5: index.erb

A Sinatra route can be used to serve static files. By default static files are served from the `./public` directory that is located under in the same directory ~~with~~ the application. A Sinatra application, though ~~it is~~ minimalistic, it is not limited to default options, thus one can configure different port numbers, root directories, custom template engines and locations, etc. Other web servers similar to Sinatra are: Flask in Python, Netty in Java, and goa in golang.

2.4 LTI tool producer

This section presents a TP written in Ruby Sinatra that implements the Basic Outcomes Service of the LTI specification, and is integrated into the Canvas LMS which will act as a TC. The TP has three routes (listed in Table 2.1).

Table 2.1: Routes of the TP

`launch`	route for launching the external tool
`assignment`	route for starting an assignment
`report`	route for reporting the result of the assignment to Canvas LMS

The `launch` route implements the LTI Launch functionality of the LTI specification, accepts requests for launching the external tool, and initiates a unique session per request. The `assignment` route checks for a valid session, and then returns an HTTP response with an HTML form. The form is the assignment and in this example contains a simple arithmetic question that the student has to reply to by submitting her answer in the form's input. Finally, the `report` route validates the student's input, and reports a pass/fail grade to the TC.

This example assumes that a Canvas instructor has created an assignment and configured it to launch the TP. The following code snippets present the code implementation of the TP (inspired by lti_example from this github repository[25] of Instructure Inc.), the functionality of each route, and the XML messages that are used to communicate with the TC.

Listing 2.6 shows the code dependencies to implement the TP. First it requires the `sinatra` gem* and the `oauth` gem (used to implement the service provider, according to the LTI specification for authorization between a a TP and a TC). The `$oauth_keyt` and `$oauth_secret` variables define the key and secret that is used by the TP to identify the TC. These variables are configured in a Canvas LMS when specifying the external tool. Finally the `disable :protection` statement allows for the HTML content produced by the Sinatra application to be embedded into an HTML frame of the TC, and the `enable :sessions` statement allows ~~for~~ session information to be used between subsequent HTTP requests to Sinatra routes.

```
# dependencies
require 'sinatra'
require 'oauth'
require 'oauth/request_proxy/rack_request'

# key and secret for authenticating requests from the TC
$oauth_key = "test"
$oauth_secret = "secret"

# disable x-frame to allow embedding the TP in the TC
disable :protection

# ennable sessions for uniquely identifying students
enable :sessions
```

Listing 2.6: Code dependencies and some global variables of the TP

The `launch` route shown in Listing 2.7 is responsible for authorizing a request from the TC to launch the assignment. First it verifies the `request` against the `secret` variable. If the authorization fails, then a text message is returned to inform the Canvas user that the integration of the tool was not successful. After the authorization succeeds, the HTTP request parameters `lis_outcome_service_url` and `lis_result_sourcedid` (these correspond to the LTI LIS services) are read. The first corresponds to the TC URL that is used to report a grade for an assignment, while the latter is a unique identifier that is used to map an assignment grade for a particular student. If these parameters were not provided when Canvas invoked this route, then the request will fail. By default Canvas sets these parameters when a tool provider is correctly configured as graded assignment. After the successful verification of the afore mentioned parameters, their values are stored in corresponding session objects and the route redirects to the `get /assignment` route.

```
post "/launch" do
  # verify the request of the TC
  begin
```

*Ruby gems are versioned packages of ruby source code. In practice they are libraries that are hosted in public servers that make them available for download via ruby package management systems.

```
    signature = OAuth::Signature.build(request, :
   consumer_secret => $oauth_secret)
    signature.verify() or raise OAuth::Unauthorized
  rescue OAuth::Signature::UnknownSignatureMethod,
         OAuth::Unauthorized
    return %{Unauthorized attempt. Make sure you used the
   consumer secret "#{$oauth_secret}"}
  end

  # Verify that this is a valid request to perform an
   assignment
  unless params['lis_outcome_service_url'] && params['
   lis_result_sourcedid']
    return %{It looks like this LTI tool was not launched as
   an assignment, or you are trying to do the assigment as a
   teacher rather than as a a student.}
  end

  # store the relevant parameters from the launch into the
   user's session, for
  # access during subsequent HTTP requests.
  %w(lis_outcome_service_url lis_result_sourcedid).each { |v|
   session[v] = params[v] }

  # Go to the assignment
  redirect to("/assignment")
end
```

Listing 2.7: Launch route

The `/assignment` route, presented in Listing 2.8, starts by validating the session variable `lis_result_sourceid`. If this parameter was not set, then the tool was not launched via the TC, hence an error text message is returned. This error message will be visible to the user's browser (either as a frame within Canvas LMS or as a new tab ~~on~~ the user's browser). If the session is valid, then the route replies with an HTML form that is rendered by the user's browser. This form includes a simple arithmetic addition question and an input field for the student to reply. The form action sends the form to the `report` route ~~and~~ using HTTP `post` method. When the student presses the submit button within the browser, the `report` route is invoked. Note that in this listing the form ~~ha~~ been included directly in the route block, but it could have been placed in a ruby template ~~such as the one of~~ Listing 2.5.

```
get "/assignment" do
  # Verify the validity of the session
  unless session['lis_result_sourcedid']
    return %{You need to take this assignment through Canvas.}
  end

  # Render a form with the assignment question.
```

```
  <<-HTML
  <html>
    <head><title>Demo LTI Assignment</title></head>
    <body>
      <form action="/report" method="post">
        <p>What is the sum of 100 + 200 ?</p>
        <input name='sum' type='text' width='5' id='sum'
   required />
        <input type='submit' value='Submit' />
      </form>
    </body>
  </html>
  HTML
end
```

Listing 2.8: Assignment route

> I found it informative to do a Wireshark capture of the tcp traffic to/from the tcp.port==xxxx (where xxxx is the port you are using) so that one can see the traffic between Canvas and Sinatra. If this is done over HTTP you can easily see what is passed. I was running the local Sinatra on my local host, so I just looked at the loopback address and the port I set my LTI external tool to use.

The `report` route is displayed in Listing 2.9 and is invoked when the student submits the form. If the form parameter `sum` is not provided, then the user is redirected (again) to the assignment via the corresponding route. Upon successful validation of the form input, an XML response message is defined and sent to Canvas via the appropriate LIS services to report the student's grade for this assignment. The format of the XML message is based upon the `imsx_POXEnvelopeRequest` class defined in the XML schema of the IMS General Web Services documentation [26] and described in the LTI 1.0 implementation guide [27].

The body of the message contains the field `sourceID` that is assigned the value of the session variable `#session['lis_result_sourcedid']`~~.~~, and the resultScore field that corresponds to the assignment's grade ~~and gets~~ the value 1 in the `textString` subfield if the provided sum was 300 or 0 otherwise. The corresponding assignment had been configured earlier in Canvas to accept a maximum of 1 point for the grade for this assignment.

The message is signed according to OAuth 1.0 protocol* using the same consumer key and secret that were provided during the LTI launch request (`launch` route). The message is posted synchronously to the Canvas LIS service defined by `session['lis_outcome_service_url']` using a Multipurpose Internet Mail Extensions (MIME)† encoding, and the response is stored in the `response` variable.

*OAuth provides a method for clients to access server resources on behalf of a resource owner (such as a different client or an end-user). It also provides a process for end-users to authorize third-party access to their server resources without sharing their credentials (typically, a username and password pair), using user-agent redirections.[28]

†The MIME-type is a two-part identifier for file formats and format of contents transmitted on

Because the post was done synchronously the code will wait until the response to this post is received. Thus the body of the response can be used to compute the message to be displayed to the user via their browser.

```
post "/report" do
  sum = params['sum']
  if !sum || sum.empty?
    redirect to("/assignment")
  end

  # now post the score to canvas. Make sure to sign the POST
   correctly with
  # OAuth 1.0, including the digest of the XML body. Also make
    sure to set the
  # content-type to application/xml.
  xml = %{
<?xml version = "1.0" encoding = "UTF-8"?>
<imsx_POXEnvelopeRequest xmlns = "http://www.imsglobal.org/lis
   /oms1p0/pox">
  <imsx_POXHeader>
    <imsx_POXRequestHeaderInfo>
      <imsx_version>V1.0</imsx_version>
      <imsx_messageIdentifier>12341234</imsx_messageIdentifier
   >
    </imsx_POXRequestHeaderInfo>
  </imsx_POXHeader>
  <imsx_POXBody>
    <replaceResultRequest>
      <resultRecord>
        <sourcedGUID>
          <sourcedId>#{session['lis_result_sourcedid']}</
   sourcedId>
        </sourcedGUID>
        <result>
          <resultScore>
            <language>en</language>
            <textString>#{sum == 300 ? 1 : 0}</textString>
          </resultScore>
        </result>
      </resultRecord>
    </replaceResultRequest>
  </imsx_POXBody>
</imsx_POXEnvelopeRequest>
  }
  consumer = OAuth::Consumer.new($oauth_key, $oauth_secret)
  token = OAuth::AccessToken.new(consumer)
  response = token.post(session['lis_outcome_service_url'],
   xml, 'Content-Type' => 'application/xml')
```

the Internet.

```
  headers 'Content-Type' => 'text'
  %{
Your score has #{response.body.match(/\bsuccess\b/) ? "been
    posted" : "failed in posting"} to Canvas. The response was:
#{response.body}
  }
end
```

Listing 2.9: Report the assignment grade to Canvas

Lastly the contents of `reponse` are evaluated and checked whether posting the grade was successful or not, and a text message is sent to the user to be rendered by her browser informing her about the status of posting the grade to Canvas. The response of a successful post is highlighted in Listing 2.10 in the `imsx_codeMajor` xml field.

```
<?xml version="1.0" encoding="UTF-8"?>
<imsx_POXEnvelopeResponse xmlns="http://www.imsglobal.org/
    services/ltiv1p1/xsd/imsoms_v1p0">
  <imsx_POXHeader>
    <imsx_POXResponseHeaderInfo>
      <imsx_version>V1.0</imsx_version>
      <imsx_messageIdentifier/>
      <imsx_statusInfo>
        <imsx_codeMajor>success</imsx_codeMajor>
        <imsx_severity>status</imsx_severity>
        <imsx_description/>
        <imsx_messageRefIdentifier>12341234</
    imsx_messageRefIdentifier>
        <imsx_operationRefIdentifier>replaceResult</
    imsx_operationRefIdentifier>
      </imsx_statusInfo>
    </imsx_POXResponseHeaderInfo>
  </imsx_POXHeader>
  <imsx_POXBody><replaceResultResponse/></imsx_POXBody>
</imsx_POXEnvelopeResponse>
```

Listing 2.10: XML response from Canvas

2.4.1 Integration of an external application into Canvas LMS

The text above presented how to develop a simple LTI producer that supports graded assignments. The Graphical User Interface (GUI) of Canvas LMS allows the integration of external applications via different options such as manual configuration forms, launch URLs, and pasting XML entries. This section will present how to configure the external tool of the previous section using a manual configuration form via the `Settings->Apps->External Apps->Add App` menu for a course. Here we assume that an instructor wishes to add an external app for a particular course. The input form shown in Figure 2.3 is loaded. The instructor inputs a name for the application, the LTI Launch URL, and the consumer key and secret.

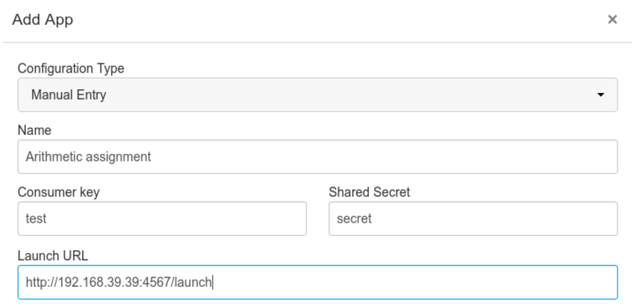

Figure 2.3: Adding an external application to Canvas

After adding this external tool, the instructor creates a new assignment, configures it to launch the application within Canvas, or using an external window as shown in Figure 2.4, and specifies a grading scheme. Once the assignment is configured and published in Canvas, a student can do this assignment via the course page. Section 2.5 explains how to integrate external applications using URLs and XML configuration.

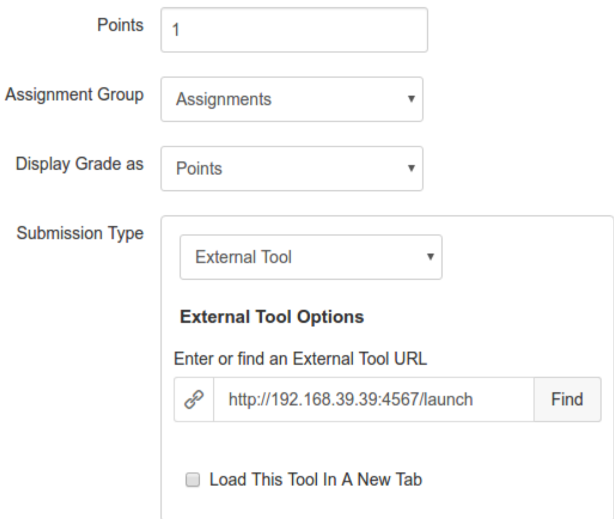

Figure 2.4: Configuring an assignment to use an external tool

2.4.2 Securing the connection between a TP and a TC

The communication between Canvas LMS and external application tools is by default expected to be performed using the Hypertext Transfer Protocol Secure

(HTTPS)* protocol. In the example presented in previous section, the communication between the TP and the TC was over HTTP, hence Canvas generated a corresponding while launching the TP. The Sinatra web-server can be easily configured to listen for HTTPS connections on some port. Listing 2.11 shows such configuration of the Sinatra web server (named Webrick). HTTPS requires a TLS certificate which for the purposes of this example was issued and signed using the OpenSSl[29] cryptography and TLS toolkit, rather than a trusted third party Certificate Authority (CA).

*HTTPS is a protocol for communication over HTTP within a connection encrypted by Transport Layer Security (TLS). TLS uses a public and a private encryption key to generate a session key which is used to encrypt the data flow between client and server. An HTTP message is encrypted prior to transmission and decrypted upon arrival.

```
require 'sinatra/base'
require 'webrick'
require 'webrick/https'
require 'openssl'

CERT_PATH = '/opt/CA/'

webrick_options = {
  :Port               => 8443,
  :Logger             => WEBrick::Log::new($stderr, WEBrick::
   Log::DEBUG),
  :DocumentRoot       => "/ruby/htdocs",
  :SSLEnable          => true,
  :SSLVerifyClient    => OpenSSL::SSL::VERIFY_NONE,
  :SSLCertificate     => OpenSSL::X509::Certificate.new(File.
   open(File.join(CERT_PATH, "cert.pem")).read),
  :SSLPrivateKey      => OpenSSL::PKey::RSA.new(File.open(File
   .join(CERT_PATH, "key.pem")).read),
  :SSLCertName        => [ [ "CN", '127.0.0.1' ] ]
}

class MyServer  < Sinatra::Base
    post '/' do
      "Hellow, world!"
    end
end

Rack::Handler::WEBrick.run MyServer, webrick_options
```

Listing 2.11: TLS configuration of a Sinatra application

Listing 2.12 shows how to generate a TLS certificate using the OpenSSL command line tool. The command is `openssl req` and it takes several arguments such as `-new` (request new certificate), `-x509` (format of the public key), `-extensions v3_ca` (the extensions to add for a self signed certificate, shown in the corresponding block of Listing ??, `-keyout key.pem` (the output file for storing the key), `-out cert.pem` (the output file for storing the self-signed certificate), `-days 365` (the number of days until the certificate expires), and finally the sample configuration file `openssl.conf` for reading the default values.

```
openssl req -new -x509 -extensions v3_ca -keyout key.pem -out
   cert.pem -days 365 -config ./openssl.conf
```

Listing 2.12: Generating a self signed TLS certificate and encryption key

The OpenSSL configuration shown in Listing 2.13, is a sample file containing default values for generating a TLS certificate and a public key file, and is available for download in Markus Redivo's page[30] "Creating and Using SSL Certificates". More details regarding the use of the **req** command of the OpenSSL toolkit can be found in the corresponding man page[31], and information about the configuration file can be found in Phil Dibowitz's blog page[32].

```
---Begin---
# OpenSSL configuration file.

# Establish working directory.
dir = .

[ ca ]
default_ca     = CA_default

[ CA_default ]
serial         = $dir/serial
database       = $dir/index.txt
new_certs_dir  = $dir/newcerts
certificate    = $dir/cacert.pem
private_key    = $dir/private/cakey.pem
default_days   = 365
default_md     = md5
preserve       = no
email_in_dn    = no
nameopt        = default_ca
certopt        = default_ca
policy         = policy_match

[ policy_match ]
countryName            = match
stateOrProvinceName    = match
organizationName       = match
organizationalUnitName = optional
commonName             = supplied
emailAddress           = optional

[ req ]
default_bits       = 1024     # Size of keys
default_keyfile    = key.pem  # name of generated keys
default_md         = md5      # message digest algorithm
string_mask        = nombstr  # permitted characters
distinguished_name = req_distinguished_name
req_extensions     = v3_req

[ req_distinguished_name ]
# Variable name          Prompt string
#----------------------   ----------------------------------
```

```
0.organizationName       = Organization Name (company)
organizationalUnitName   = Organizational Unit Name (department, division)
emailAddress             = Email Address
emailAddress_max         = 40
localityName             = Locality Name (city, district)
stateOrProvinceName      = State or Province Name (full name)
countryName              = Country Name (2 letter code)
countryName_min          = 2
countryName_max          = 2
commonName               = Common Name (hostname, IP, or your name)
commonName_max           = 64

# Default values for the above, for consistency and less typing.
# Variable name                Value
#----------------------       ------------------------------
0.organizationName_default  = The Sample Company
localityName_default        = Metropolis
stateOrProvinceName_default = New York
countryName_default         = US

[ v3_ca ]
basicConstraints       = CA:TRUE
subjectKeyIdentifier   = hash
authorityKeyIdentifier = keyid:always,issuer:always

[ v3_req ]
basicConstraints     = CA:FALSE
subjectKeyIdentifier = hash

----End----
```

Listing 2.13: Sample OpenSSL configuration for issuing SSL/TLS certificates

2.5 LTI applications

Edu App Center[33] is an open database for learning tools maintained by Instructure [34] and among its several services, it offers a collection of open learning applications that implement the LTI specification~~, and~~ can be integrated with different LMSs. The user can apply ~~several~~ filters to locate an appropriate tool and can browse tutorials about integrating a tool with the LMS of their choice. Often these tools are hosted by third party services (e.g GitHub, Youtube, Turnitin). The goal of Edu App Center is to enable instructors to easily configure these external applications ~~to~~ their courses, thus providing and fostering a market place for LTI applications.

Section 2.4.1 presented how an instructor can integrate a Ruby Sinatra external application into Canvas LMS using web form. This approach is limited to the functionality of Canvas LMS. ~~The~~ alternative method for integrating external applications via XML configuration can be used across different LMSs. Edu App

Center offers such configurations for every LTI tool listed in the marketplace. Additionally, it provides the XML Config Builder service, that allows instructors to generate XML for integrating custom built external LTI applications into different LMSs. Listing 2.14 shows an example of such XML entry (generated by the Edu App Center's XML Config Builder) that was used to integrate the Ruby Sinatra application (presented in the previous section) into Canvas.

First, the XML version and the charset encoding are defined. Then the `cartridge_basiclti_link xmlns` specifies that this is an LTI link that can be used for integrating an external application. This block contains the whole XML configuration. It starts by defining the IMS Global XML schema that is used to describe this entity. Then the LTI Launch URL is specified (`blti:launch_url`), and it is followed by metadata, regarding the title (`blti:title`) and description (`blti:description`) of the external application. Finally, it defines a block for ~~lti~~ extensions (`blti:extensions platform`) that specifies the LMS platform ~~to~~ act as a TC for this TP. This block of XML code can contain information that is specific to each LMS that is supported by the TP.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<cartridge_basiclti_link xmlns="http://www.imsglobal.org/xsd/
    imslticc_v1p0"
  <!-- Definition of the XML Schema -->
  xmlns:blti = "http://www.imsglobal.org/xsd/imsbasiclti_v1p0"
  xmlns:lticm ="http://www.imsglobal.org/xsd/imslticm_v1p0"
  xmlns:lticp ="http://www.imsglobal.org/xsd/imslticp_v1p0"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation = "http://www.imsglobal.org/xsd/
   imslticc_v1p0 http://www.imsglobal.org/xsd/lti/ltiv1p0/
   imslticc_v1p0.xsd
  http://www.imsglobal.org/xsd/imsbasiclti_v1p0 http://www.
   imsglobal.org/xsd/lti/ltiv1p0/imsbasiclti_v1p0.xsd
  http://www.imsglobal.org/xsd/imslticm_v1p0 http://www.
   imsglobal.org/xsd/lti/ltiv1p0/imslticm_v1p0.xsd
  http://www.imsglobal.org/xsd/imslticp_v1p0 http://www.
   imsglobal.org/xsd/lti/ltiv1p0/imslticp_v1p0.xsd">

  <!-- The LTI Launch url -->
  <blti:launch_url>http://192.168.39.39:4567/launch</
   blti:launch_url>
  <!-- Title of the External Application -->
  <blti:title>Arithmetic Assignment</blti:title>
  <!-- Description for the external application -->
  <blti:description>Sample arithmetic assignment tool</
   blti:description>
  <-- Configuration specific to the TC -->
  <blti:extensions platform="canvas.instructure.com">
    <lticm:property name="privacy_level">public</
   lticm:property>
  </blti:extensions>
</cartridge_basiclti_link>
```

Listing 2.14: I am a comment

Should mention apps that can be used for evaluation of the internetworking assignments.

2.6 Previous efforts to provide on-line exercise material

Traditional practice events in Computer Science involve laboratory environments and exercises based on virtual hardware and domain specific software. One of the problems is creating and managing these environments. Previously such material was packaged in virtual machines or run in an isolated environment (such as a sandbox or linux container as will be described in Section 2.7).

With the rapid growth of e-learning courses, the need for on-line exercise material has grown. Efforts in fields of cybersecurity include ”A Comparison of

Virtual Lab Solutions for Online Cyber Security Education"[35], "Top 10 Hands-on Cybersecurity Exercises"[36],

TODO: extend this paragraph

In addition to the environment, there is a need for domain specific source material. Some useful references and sources for exercise material regarding networking include "Hands-On Experience to a Massive Open Online Course on openHPI"[4], "Some Experiences in Using Virtual Machines for Teaching Computer Networks"[3], and "V-Lab: A Mobile Virtual Lab for Network Security Studies"[37].

2.6.1 IK1552

More info

- What are the requirements for successfully completing an exercise?
- What information about a student's progress in or success with the exercise should (or could) be communicated back to the LMS?

  I would expect to get back:
  1. a score (either scaler or vector)
  2. time spent in exercise and the date & time when the exercise was completed
  3. potentially file containing the results of the exercise (this could be text, pcap file, etc.)

- How are these requirements met by the chosen technologies?

2.7 Linux Containers

A container is a light weight operating system running inside the host system, executing instructions native to the ~~core~~ CPU, eliminating the need for instruction level emulation or just in time compilation [38]. Linux Containers (LXC)[39] is an operating-system-level virtualization method for running multiple isolated Linux systems (containers) on a host using a single Linux kernel. Its purpose is to virtualize a single application rather than a whole operating system inside a virtual machine. LXC uses cgroups* to isolate resources (such as CPU, memory, network, etc.) and namespaces† to isolate the application from the operating system [41].

Docker [42] is a Linux container engine, that provides the ability to manage containers as self contained images. Docker utilizes LXC for the container implementation, has image management capabilities, and implements a Union File System (UnionFS). It features resource isolation via cgroups and namespaces,

*Control groups (cgroups) is a Linux kernel feature that is responsible for managing resources such as CPU, memory, disk I/O, network, etc.

†A namespace wraps a global system resource (process IDs, mount points, network devices, network stacks, ports, etc.) in an abstraction that makes it accessible to the processes. Within a namespace each process has its own isolated instance of the global resource. Changes to the global resource are visible to other processes that are members of the namespace, but are invisible to other processes [40].

network and file system isolation through LXC functionality, and allows managing the lifecycle of a container. [3[illegible]lthough docker initially utilized LXC as the only execution driver for resource isolation, lately it introduced libcontainer [43], which includes its own implementation for resource isolation, but also bindings to leverage other technologies such as LXC, libvirt-lxc[44] and systemd-nspawn[45], thus ~~being~~ a cross-system abstraction layer for packaging, delivering and running applications in isolated environments. The implementation and functionality of libcontainer is defined by the Open Container Initiative (OCI)[46] specification which defines the image formats, the image management interface, and the container runtime life-cycle.

Docker leverages a client-server architecture. The server is called docker daemon, and it is responsible for container runtime. It has capabilities for building, running, and distributing docker containers. The client~~,~~ is a user interface for communicating with the docker daemon. The client has several implementations, including a command line tool [47]~~,~~ and the Docker Remote API [48]. The ecosystem ~~of Docker~~ includes different technologies and tools for managing images, container and application runtime, infrastructure deployment and orchestration, etc. The Docker Hub is an image registry~~,~~ that stores container images in a ~~way~~ similar ~~to~~ traditional package management ~~of~~ software artifacts. An image is part of a repository~~, has~~ an author and a version, thus making easy to distribute and discover~~ an image configuration~~.

Listing 2.15 illustrates how a container image can be downloaded from the Docker Hub using the command line interface of the docker daemon. The command `docker pull ubuntu:14.04` requests ~~to~~ download the image of ~~the~~ ubuntu repository that is tagged with version 14.04. Docker daemon connects to the Hub, requests ~~for the~~ particular image of that repository, and starts downloading the image configuration and dependencies. Finally, after ~~it finishes~~ downloading, ~~it~~ creates a hash string of the image using the Secure Hash Algorithm (SHA) algorithm~~, which~~ is used to identify ~~uniquely~~ the image in the local registry ~~of~~ docker daemon.

```
$: docker pull ubuntu:14.04
14.04: Pulling from library/ubuntu

ba76e97bb96c: Pull complete
4d6181e6b423: Pull complete
4854897be9ac: Pull complete
4458f3097eef: Pull complete
9989a8de1a9e: Pull complete
Digest: sha256:062bba17f92e749bd3092e7569aa0\
    6c6773ade7df603958026f2f5397431754c
Status: Downloaded newer image for ubuntu:14.04
```

Listing 2.15: ~~command~~ docker pull

Using the command line client, docker can list all downloaded images along with a set of metadata for these images. Listing 2.16 shows the output of command `docker images`, which contains the name of the repository, the repository tag, a unique identifier of the image, ~~and~~ additional information such as when the image was created and stored in the Docker Hub~~,~~ and its size.

```
$: docker images
REPOSITORY    TAG       IMAGE ID        CREATED       SIZE
ubuntu        14.04     4d44acee901c    3 days ago    187.9 MB
```

Listing 2.16: ~~command~~ docker images

The container runtime~~,~~ defines the different states of a container~~, which are~~ created, started, paused, stopped, and deleted. In order to run an application inside an isolated environment, first a container has to be created from an existing image~~,~~ and then started. Listing 2.17 shows the command `docker run`~~.~~ which specifies ~~to run~~ a container from a particular image and execute ~~an~~ application~~.~~

```
$: docker run -t -i ubuntu:14.04 /bin/bash
```

Listing 2.17: ~~command~~ docker run

In more detail, ~~it~~ creates a container from the image `ubuntu:14.04`, and configures ~~it~~ according to the specified arguments. The command argument `-t` ~~requires to~~ allocate ~~a pty~~ pseudoterminal [49], and the argument `-i` ~~requires to~~ attach the standard input to ~~the~~ pseudoterminal. Finally ~~it starts~~ the container and executes the command `\bin\bash`.

Listing 2.18 illustrates to `docker ps` command which lists the containers that are in running state. The output of the command includes information such as the unique identifier of the container, the container image, the command that is running, and other information such as when the container was created it, when it started running, what port bindings the container has with the host operating system, and a unique name.

```
$: docker ps
CONTAINER ID   IMAGE          COMMAND        CREATED         STATUS        PORTS   NAMES
91af84830636   ubuntu:14.04   "/bin/bash"    3 seconds ago   Up 2 seconds
      lonely_lichterman
```

Listing 2.18: ~~command~~ docker ps

The commands presented previously are ~~just~~ a subset of the command line interface of the docker client. Figure 2.5, ~~which if listed in~~ the documentation ~~of~~ the Docker Remote API, shows a state diagram of a container, along with the various commands and events that are responsible for containers transitioning between different states.

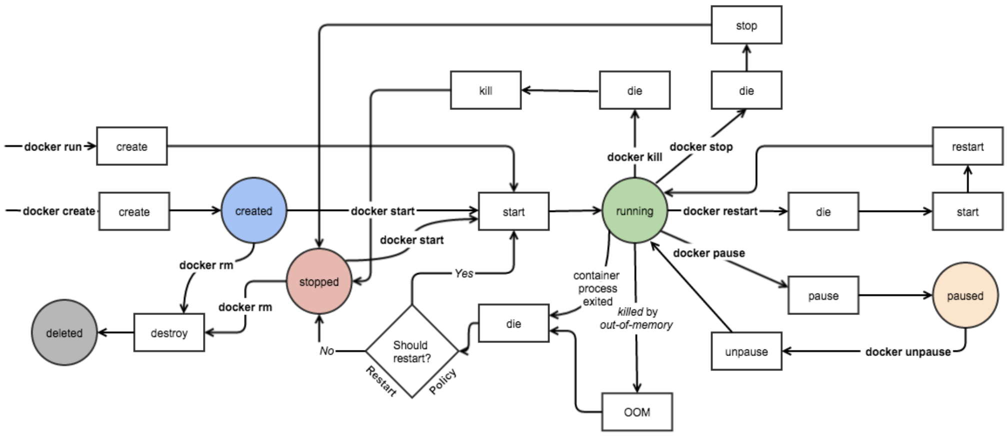

Figure 2.5: States of the container lifecycle

Listing 2.17 showed how to run the bash shell process inside a linux container. The code snippets of listings 2.19 and 2.20 illustrate how one can install a package in the operating system of the container~~,~~ and create a new image ~~out of it~~.

```
root@91af84830636:/# apt-get install traceroute
```

Listing 2.19: Installing a package in the container Operating System

Using the apt package manager of Ubuntu, the root user installs the traceroute package. Then this running container is used to create a new image, that will contain the ~~changes performed in the previous step~~.

```
$: docker commit -m "traceroute-package" -a "KTH" 91af84830636
    my-ubuntus:traceroute
```

Listing 2.20: Create a new docker image out of a running container

The command `docker commit` accepts a `-m` parameter ~~for~~ a commit message, a `-a` parameter ~~to~~ specify the author of this commit, the id of the container that will be used to create a new image (in this case 91af84830636), the name of the repository (my-ubuntu) and the reference tag for this repository (:traceroute). Executing the command `docker images` as shown in listing 2.21, will verify that the image was created.

```
$: docker images
REPOSITORY   TAG          IMAGE ID       CREATED         SIZE
ubuntu       14.04        4d44acee901c   3 days ago      187.9 MB
my-ubuntus   traceroute   1261c79eb3da   4 seconds ago   166.9 MB
```

Listing 2.21: List the ~~newly created~~ docker image

Linux containers can be used to create pre-configured machines for laboratory assignments of an Internetworking course. By creating container images tailored to the needs of each assignment, a student can focus on the exercise ~~and not on the~~ details that are not relevant to the learning process. A software solution that supports ~~functionality such as~~ creating images and running containers on demand~~,~~ can be very useful for e-learning, ~~where~~ a student ~~in~~ just a few seconds ~~get~~ access ~~to~~ a unique laboratory environment via her web browser.

2.8 Web based shell emulators

When it comes to e-learning assisted by LMSs, students are used to perform most of their learning tasks via their web browser. Using pre-configured laboratory environments based on docker images entails the same risks ~~of~~ traditional labs, ~~where~~ the student ~~will have~~ to install docker~~,~~ and manually execute a series of commands before she will be able to ~~start~~ focus~~ing~~ on the learning process. An alternative solution would be to support such environments in a remote server, ~~and~~ provide ~~access to~~ the student via a web browser. The software that provides access to a linux shell via a web browser is often called web based terminal emulator. The technology that provides communication between the server (terminal emulator) and the client (web browser) is called Web-based Secure Shell (SSH). The server side implementation involves a web application~~,~~ that ~~is~~ accept~~ing~~ requests for keyboard events and forwards ~~them~~ to a secure shell client communicating with the connected SSH server. The 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 [50].

There are several implementations of web based shell emulators ~~among which are~~ GateOne[51] and Shell In A Box[52]. The latter, implements a web server that can export arbitrary command line tools to a web based terminal emulator. This emulator is accessible to any JavaScript and CSS enabled web browser. The server listens on a specified port~~,~~ and publishes services that are displayed by a VT100[53] emulator implemented as an Asynchronous JavaScript and XML (AJAX)[54] web application.

2.9 Related work

The support for Interoperability specifications by several LMSs has allowed rapid experimentation and implementation of external application frameworks that offer a variety of on-line training events for various Computer Science courses. This section presents some of these frameworks and describes how they are relevant to this project.

2.9.1 EDURange

Designing on-line training environments for the field of cyber security requires overcoming some technical constraints, such as high availability and scalability, and pedagogical limitations, such as teaching analysis skills to understand complex systems and concepts via practicing [2]. EDURange addresses these issues by designing an open source framework that provides interactive security exercises in an elastic cloud environment [55].

EDURange is a software framework, designed to work on Amazon Elastic Compute Cloud (EC2) [56]. It allows teachers to easily build and scale dynamic virtual environments to host cybersecurity training [57]. This framework provides ease of use for instructors, by offering the flexibility to specify exercises at a high level and allowing the instructor to configure different aspects of the training scenarios in order to provide a tailored learning experience that focuses on analysis skills.

2.9.2 GLUE!

Group Learning Uniform Environment (GLUE!) is a middle-ware integration architecture that aims to standardize the integration of existing external learning tools into several LMSs [58]. It facilitates the instantiation and enactment of collaborative learning situations within LMSs, by using the distinctive administrative features of these systems to manage users and groups. LTI ~~or~~ Sharable Content Object Reference Model (SCROM) are two specifications for the integration of external learning tools into an LMS. ~~E~~ach LMS usually supports a single interoperability specification; thus, developing a universal external tool requires a substantial development effort to support the different standards. In contrast, GLUE! proposes a software architecture that takes advantage of the common integration features of LMSs to integrate multiple existing learning tools into multiple LMSs.

2.9.3 INGInious

Programming exercises are the most common form of practice for students learning computer science. Traditionally, the evaluation of these exercises, requires grading of reports, reading source code, and testing source code, thus making it time consuming, especially for large classes (i.e., large numbers of students). INGInious [7, 59, 60, 61] is a software framework that empowers instructors to easily construct coding tasks and it supports automatic evaluation and grading of the code, thus providing both students and teachers with constructive feedback.

The framework consists of two main components: the frontend and the backend. The frontend provides a web interface where students perform programming tasks and an administration module that allows instructors to design these tasks. The backend is responsible for running and grading the code inside remote isolated Linux containers. Each container is specifically built for a particular programming language, according to configuration provided by the instructor or the administrator of the system, thus supporting the evaluation of tasks written in any programming language that runs in a Linux environment.

One of the main features of INGInious is that the frontend component can be used either as a stand-alone web application or as an external learning tool that is integrated into an LMS using the LTI specification. Additionally, the backend component scales horizontally very easily, since it utilizes a docker container for every task request, therefore it is suitable for MOOC platforms.

A programming task in INGInious is designed using a configuration file (`task.yaml`) that identifies the problem to be solved by the student, and the evaluation process, a template file (template.py) that presents the task to the student, and defines the input field for the code, and finally, a file (`run`) that executes the student code, and validates the output. The following code samples show the minimum configuration required by the instructor, to design a simple "Hello World" task in Python. Listing 2.22 is the `task` file. It starts with key-value pairs that are used to describe the `name` and `context` of the task.

```yaml
name: "Hello World!"
context: "In this task, you will have to write a python script
    that displays 'Hello World!'."
problems:
  question1:
    name: "Let's print it"
    header: "def func():"
    type: "code"
    language: "python"
limits:
  time: 10
  memory: 50
  output: 1000
environment: "default"
```

Listing 2.22: Definition of a task in task.yaml

Then it defines the `problems` that have to be solved to complete this task. Each problem has a unique name within the task (`question1`) and a series of metadata such as the programming language to be used for solving the problem, and the text input to print in the input form. Finally it contains other metadata that defines the resources of the virtual environment that will be used to evaluate the code.

```python
def func():
  @ @question1@@

  func()
```

Listing 2.23: Code input of question1 in template.py

Listing 2.23 defines the input field ~~in~~ which the student will ~~have to~~ input ~~the~~ code. Finally, the `run` file defined in Listing 2.24, is a shell script, that parses the input code using the INGInious commands `parsetemplate`, then evaluates the expected output against the results of the input function using the command `run_student`. Finally it prepares the result of the task using the `feedback` command.

```bash
#! /bin/bash

# Parse the template and put the result in studentcode.py
parsetemplate --output studentcode.py template.py

# Verify the output of the code...
output=$(run_student python studentcode.py)
if [ "$output" = "Hello World!" ]; then
  # The student succeeded
  feedback --result success --feedback "Success!"
else
  # The student failed
  feedback --result failed --feedback "Your output is $output"
```

```
fi
```

sting 2.24: Evaluation of student code by the run file

Detailed information for specifying a task in INGInious platform can be found in the official teacher documentation [62]. As part of the research ~~process of~~ this project, the LTI component of INGInious was configured with Canvas LMS, to perform sample programming tasks like the "Hello World!" that was explained earlier.

* Should reference the installation documentation page of INGInious, and the configuration page of LTI. * Can reference the Vagrant file that provisions the INGInious environment, as part of replicating the methodology.

Mention the python implementation
mention that it can only evaluate the output of programms running in the containers

2.10 Summary

It is nice to bring this chapter to a close with a summary. For example, you might include a table that summarizes the ideas of others and the advantages and disadvantages of each ? so that later you can compare your solution to each of these. This will also help guide you in defining the metrics that you will use for your evaluation.

Chapter 3

Methodology

What scientific or engineering methodology are you going to use and why have you chosen this method. What other methods did you consider and why did you reject them. What are your goals? (What should you be able to do as a result of your solution - which could not be done well before you started?) What you are going to do? How? Why? For example, if you have implemented an artifact what did you do and why? How will your evaluate it.

In the design science paradigm, the rigor cycle provides past knowledge to the research project to ensure its innovation. Researchers thoroughly research and reference the relevant knowledge base. The central Design Cycle iterates between the core activities of building and evaluating the design artifacts and processes of the research [9], until the acceptance criteria, as defined in the Relevance Cycle, are met.

3.1 Research Process

This thesis project is carried out using the Design Science research method. This type of research focuses on the design and construction of IT artifacts that have utility in the real world, in this case as an application environment, and aim to improve domain-specific systems and processes. In the context of this research, the real world problem is the lack of interactive virtual laboratory environments in the form of e-learning tools. More specifically, the major problem is to integrate such tools within an LMS, to assist the learning and teaching of an on-line Internetworking course.

3.2 Research Paradigm

HTTP traffic widget from FORGE http://ict-forge.eu/wp-content/uploads/2016/01/FORGE-2015-P-D312-Final.pdf

Chapter 4

Implementation

4.1 LTI Tool Provider

4.1.1 Architecture

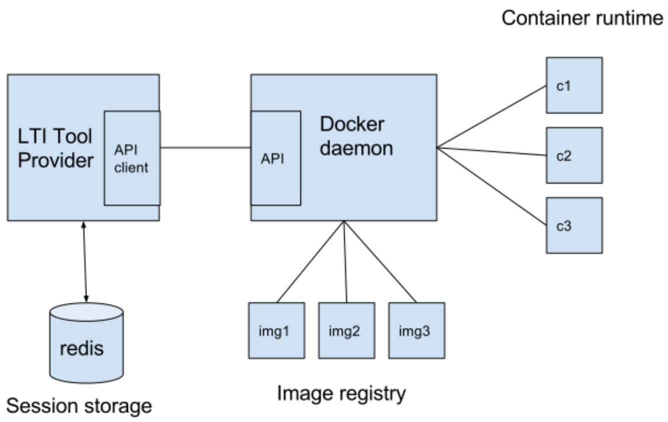

Figure 4.1: Architecture of the LTI Tool Provider

4.1.2 API Definition

4.1.3 Docker Remote API

4.1.4 Avoiding port collision

4.2 LTI Tool Client

4.2.1 Architecture

4.2.2 Admin Tool

4.2.3 Integration with Canvas

So far basic setup of the following environments has been accomplished

- Canvas LMS in a vagrant virtual environment, with admin, instructor and student accounts.
- Integration of the LTI basic outcomes service. A student takes an assignment and the result is reported as a grade back to Canvas LMS.
- Bootstrapping of the INGInious backend and frontend modules. LTI integration to LMS was not successful during the first attempts.

Chapter 5

Analysis

...

5.1 Major results

...

5.2 Reliability Analysis

...

5.3 Validity Analysis

...

5.4 Discussion

...

Chapter 6

Conclusions and Future Work

...

6.1 Conclusions

...

6.2 Limitations

...

6.3 Future work

...

6.4 Reflections

...

References

[1] William R. Watson and Sunnie Lee Watson. An argument for clarity: what are learning management systems, what are they not, and what should they become? *TechTrends*, 51(2):28–34, 2007.

[2] Stefan Boesen, Richard Weiss, James Sullivan, Michael E. Locasto, Jens Mache, and Erik Nilsen. EDURange: Meeting the Pedagogical Challenges of Student Participation in Cybertraining Environments. In *7th Workshop on Cyber Security Experimentation and Test (CSET 14)*, San Diego, CA, August 2014. USENIX Association.

[3] Ricardo Nabhen and Carlos" Maziero. *Education for the 21st Century — Impact of ICT and Digital Resources: IFIP 19th World Computer Congress, TC-3, Education, August 21–24, 2006, Santiago, Chile*, chapter Some Experiences in Using Virtual Machines for Teaching Computer Networks, pages 93–104. Springer US, Boston, MA, 2006.

[4] Christian Willems, Johannes Jasper, and Christoph Meinel. Introducing hands-on experience to a massive open online course on openhpi. In *Teaching, Assessment and Learning for Engineering (TALE), 2013 IEEE International Conference on*, pages 307–313. IEEE, 2013.

[5] Instructure, Inc. Canvas learning management system. `https://www.canvaslms.com/`. [Online; accessed 2016-02-21].

[6] Daniela Fonte, Daniela da Cruz, Alda Lopes GanÃ§arski, and Pedro Rangel Henriques. A flexible dynamic system for automatic grading of programming exercises. In *OASIcs-OpenAccess Series in Informatics*, volume 29. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[7] Guillaume Derval, Anthony Gego, Pierre Reinbold, Benjamin Frantzen, and Peter Van Roy. Automatic grading of programming exercises in a MOOC using the INGInious platform.

[8] Ricardo Queirós and José Paulo Leal. Programming exercises evaluation systems - An interoperability survey. In *CSEDU (1)*, pages 83–90, 2012.

[9] Alan Hevner and Samir Chatterjee. Design Science Research in Information Systems. In *Design Research in Information Systems*, volume 22, pages 9–22. Springer US, Boston, MA, 2010.

[10] Vijay K. Vaishnavi and William Kuechler, Jr. *Design Science Research Methods and Patterns: Innovating Information and Communication Technology*. Auerbach Publications, Boston, MA, USA, 1st edition, 2007.

[11] Alan R. Hevner. A three cycle view of design science research. *Scandinavian journal of information systems*, 19(2):4, 2007.

[12] C. Alario and S. Wilson. Comparison of the main alternatives to the integration of external tools in different platforms. In *ICERI2010 Proceedings*, 3rd International Conference of Education, Research and Innovation, pages 3466–3476. IATED, 15-17 November, 2010 2010.

[13] Open edX as an LTI Tool Provider. `https://open.edx.org/blog/open-edx-lti-tool-provider`. [Online; accessed 2016-02-28].

[14] Md. Iqbal Hossain and Md. Iqbal Hossain. Dynamic scaling of a web-based application in a cloud architecture. Master's thesis, KTH, Radio Systems Laboratory (RS Lab), 2014.

[15] Ryann K Ellis. Field guide to learning management systems, 2009.

[16] José Paulo Leal and Ricardo Queirós. A comparative study on lms interoperability. *Higher Education Institutions and Learning Management Systems: Adoption and Standardization*, page 142, 2011.

[17] Wynne Harlen and Mary James. Assessment and Learning: differences and relationships between formative and summative assessment. *Assessment in Education: Principles, Policy & Practice*, 4(3):365–379, November 1997.

[18] Janne Malfroy Kevin Ashford-Rowe. E-Learning Benchmark Report: Learning Management System (LMS) usage. `http://www.uws.edu.au/__data/assets/pdf_file/0007/452077/Griffith_UWS_Elearning_Benchmark_Report.pdf`, 2009.

[19] ISO. Information technology vocabulary. ISO 2121317 - 2382:2015, International Organization for Standardization, 2015.

[20] IMS GLOBAL Learning Consortium. Learning Tools Interoperability ®(LTI®). `http://www.imsglobal.org/activity/learning-tools-interoperability`. [Online; accessed 2016-02-23].

[21] Ricardo Queirós, José Paulo Leal, and José Paiva. Integrating rich learning applications in LMS. In *State-of-the-Art and Future Directions of Smart Learning*.

[22] IMS Learning Information Services. https://www.imsglobal.org/lis/. [Online; accessed 2016-02-28].

[23] Ruby Sinatra - official documentation page. http://www.sinatrarb.com/documentation.html. [Online; accessed 2016-07-17].

[24] Alan Harris and Konstantin Haase. *Sinatra: Up and Running*. O'Reilly Media, Inc., 1st edition, 2011.

[25] LTI Outcome Service Example using Canvas LMS. https://github.com/instructure/lti_example. [Online; accessed 2016-04-23].

[26] IMS Global General Web Services. https://www.imsglobal.org/gws/index.html. [Online; accessed 2016-07-27].

[27] IMS Global Learning Tools Interoperability™ Implementation Guide. https://www.imsglobal.org/specs/ltiv1p1/implementation-guide. [Online; accessed 2016-07-27].

[28] The OAuth 1.0 Protocol. https://tools.ietf.org/html/rfc5849. [Online; accessed 2016-07-25].

[29] OpenSSL cryptography and SSL/TLS toolkit. https://www.openssl.org/. [Online; accessed 2016-08-07].

[30] Marcus Redivo. Creating and using SSL certificates. http://www.eclectica.ca/howto/ssl-cert-howto.php. [Online; accessed 2016-08-07].

[31] OpenSSL - official documentation of command `req`. https://www.openssl.org/docs/manmaster/apps/req.html. [Online; accessed 2016-08-07].

[32] Phil Dibowitz. Openssl.conf walkthru. https://www.phildev.net/ssl/opensslconf.html. [Online; accessed 2016-08-07].

[33] An open lti app collection. https://www.eduappcenter.com/. [Online; accessed 2016-07-11].

[34] Instructure. https://www.instructure.com/. [Online; accessed 2016-07-17].

[35] Joon Son, Chinedum Irrechukwu, and Patrick Fitzgibbons. A Comparison of Virtual Lab Solutions for Online Cyber Security Education. *Communications of the IIMA*, 12(4), 2012.

[36] Richard Weiss, Jens Mache, and Erik Nilsen. Top 10 hands-on cybersecurity exercises. *J. Comput. Sci. Coll.*, 29(1):140–147, October 2013.

[37] Yugesh Suresh Bhosale and Jenila Livingston L. M. Article: V-lab: A mobile virtual lab for network security studies. *International Journal of Computer Applications*, 93(20):35–38, May 2014. Full text available.

[38] Rajdeep Dua, A Reddy Raja, and Dharmesh Kakadia. Virtualization vs Containerization to Support PaaS. pages 610–614. IEEE, March 2014.

[39] Linux containers - lxc. `https://linuxcontainers.org/lxc/introduction/`. [Online; accessed 2016-02-28].

[40] Linux programmer's manual, overview of linux namespaces. `http://man7.org/linux/man-pages/man7/namespaces.7.html`. [Online; accessed 2016-02-28].

[41] Rami Rosen. Linux containers and the future cloud. *Linux J*, 240, 2014.

[42] Docker. `https://www.docker.com/`. [Online; accessed 2016-02-28].

[43] Libcontainer implementation. `https://github.com/opencontainers/runc/tree/master/libcontainer`. [Online; accessed 2016-11-20].

[44] Lxc container driver. `https://libvirt.org/drvlxc.html`. [Online; accessed 2016-11-20].

[45] systemd-nspawn. `https://www.freedesktop.org/software/systemd/man/systemd-nspawn.html`. [Online; accessed 2016-11-20].

[46] Open container initiative. `https://www.opencontainers.org/about`. [Online; accessed 2016-11-20].

[47] Docker command line reference. `https://docs.docker.com/engine/reference/commandline`. [Online; accessed 2016-11-20].

[48] Docker remote api. `https://docs.docker.com/engine/reference/api/docker_remote_api`. [Online; accessed 2016-11-20].

[49] Unix pseudoterminal interface. `http://man7.org/linux/man-pages/man7/pty.7.html`. [Online; accessed 2016-11-20].

[50] Web based ssh. `https://en.wikipedia.org/wiki/Web-based_SSH`. [Online; accessed 2016-11-19].

[51] Gateone. `https://github.com/liftoff/GateOne`. [Online; accessed 2016-11-20].

[52] shellinabox. `https://github.com/shellinabox/shellinabox`. [Online; accessed 2016-11-20].

[53] *100 Series Technical Manual*, 1979.

[54] Anthony T. Holdener, III. *Ajax: The Definitive Guide*. O'Reilly, first edition, 2008.

[55] Edurange: A cybersecurity competition platform to enhance undergraduate security analysis skills. `http://blogs.evergreen.edu/edurange/`. [Online; accessed 2016-02-28].

[56] Amazon elastic compute cloud (amazon ec2). `https://aws.amazon.com/ec2/`. [Online; accessed 2016-02-28].

[57] EDURange Github project. 2014. [Online; accessed 2016-02-28].

[58] Carlos Alario-Hoyos, Miguel L. Bote-Lorenzo, Eduardo Gómez-Sánchez, Juan I. Asensio-Pérez, Guillermo Vega-Gorgojo, and Adolfo Ruiz-Calleja. Glue!: An architecture for the integration of external tools in virtual learning environments. *Computers & Education*, 60(1):122–137, 2013.

[59] Inginious by universitÃ© catholique de louvain. `http://inginious.org/`. [Online; accessed 2016-02-28].

[60] Github repository of inginious. `https://github.com/UCL-INGI/INGInious`. [Online; accessed 2016-02-28].

[61] Technical documentation of inginious. `http://inginious.readthedocs.org`. [Online; accessed 2016-02-28].

[62] Teacher documentation of inginious. `http://inginious.readthedocs.io/en/latest/teacher_documentation.html`. [Online; accessed 2016-02-28].

Appendix A

Appendix Name X

content X