Verkkoke 1.1 User Manual

J-Sim 1.3

Brite 2.1

Table of Contents

1 Introduction

Verkkoke 1.1 is an online teaching environment for telecommunications software and routing, based on "learning by doing" philosophy. It is designed for use in conjuction with a programming assignment. The assignment introduces students to socket programming and simplified routing protocols. Students write client programs, which connect to the Verkkoke 1.1 server. The server simulates a network of routers. The student's client represents one router in the simulated network.

First, the student's client authenticates with the server. Upon successfull authentication, the server sends routing protocol messages to the student's client. The client then builds its routing table according to the received routing protocol messages. Finally, the client submits its routing table to the server.

Key features of Verkkoke 1.1:

- Creates individual simulated networks with representative topologies for the students.
- Simulates two simplified routing protocols, one distance vector protocol and one link state protocol. Sends protocol messages to the client.
- Automatically checks the submitted routing tables.
- Gives feedback to students, so that they can learn from their mistakes, fix them and resubmit the corrected solutions.
- Compatibility with modern learning management systems through its adherence to the Sharable Content Object Reference Model (SCORM) specification.
- Graphical Configuration Interface (GCI) to easily setup and maintain Verkkoke.
- Modularity and extensibility, which allows using the sophisticated network simulation engine for teaching other network protocols in addition to the routing protocols.

2 Getting Started

2.1 Installing Verkkoke

2.1.1 Prerequisites

The software depends on the following components:

- A SCORM-compliant Learning Management System (LMS), e.g. Moodle.
- A php 5.0 enabled web server, e.g. Apache.
- A Java servlet engine, e.g. Tomcat.
- A Java runtime environment, version 1.5 or later.
- The Graphviz graph visualization software. The "dot" command of Graphviz is used. This is included in many Linux distributions.

To install these components, please refer to their own instructions.

2.1.2 Installation Steps

See [Detailed Installation], page 2, for more thorough instructions.

1. Extract the contents of the package: *`tar -zxf verkkoke-1.1.tar.gz`*.
2. Copy the '`verkkoke`' root folder (and its contents) to the desired deployment location.
3. Configure your PHP enabled web server to publish files under web directory
4. Deploy the '`topologygenerator.war`' file in the verkkoke folder to your servlet engine. Refer to your servlet engine's documentation on how to deploy a WAR (Web ARchive) file. Make sure that the "$topologygeneratorURL" variable in config.php matches the URL of the deployed servlet.
5. Deploy the SCORM package to your LMS. See the README.txt file in the "scorm" folder.
6. Configure file system permissions. Web server must be able to read from and under verkkoke directory. It must also be able to write to '`data/student_data`' directory, and to files '`server/bin/simulationserver.properties`' and '`topology_generator.conf`'. Topology Generator must be able to read '`topology_generator.conf`' from verkkoke directory, write to '`data/student_data`', '`data/brite`' and '`web/topologyPics`', and to execute `dot`. Simulation Server must be able to read its own files (including '`simulationserver.properties`' written by web server) and to write its log files (if logging is required) to '`log`' directory.
7. Launch Graphical Configuration Interface by pointing your web browser to '`web/configGUI/index.php`'. Save settings once, even if you do not need to change anything, to update Simulation Server's and Topology Generator configuration files.

2.1.3 Detailed Installation

First, make sure you have installed everything needed that is needed by Verkkoke: A PHP 5.0 capable web server (e.g. Apache), a servlet engine (e.g. Apache Tomcat), a SCORM

supporting learning management system (e.g. Moodle), Java 5.0 or newer and Graphviz' dot program[1].

Extract the contents of the package: *`tar -zxf verkkoke-1.1.tar.gz`*. This should create directory '`verkkoke`' with some subdirectories ('`data`', '`log`', '`phpinc`', '`scorm`', '`server`' and '`web`'). There should also be '`default_brite_conf`', '`default_student_data.data`' and '`topology_generator.conf`' files.

2.1.3.1 Configuring Web Server

Configure your web server to publish files in '`web`' directory, including its subdirectories. Note that scripts in '`web`' directory need access to some files in '`verkkoke`' directory or its subdirectories, so '`web`' directory should not be copied or moved anywhere, neither should '`verkkoke`' directory be accessible via HTTP[2].

Web server, or more precisely the PHP engine, must also be able to write files to '`data`' directory. If Graphical Configuration Interface is used, server must also be able to write Simulation Server's and Topology Generator's configuration files ('`server/bin/simulationserver.properties`' and '`topology_generator.conf`' respectively).

Also note that the web server must be able to handle PHP 5.0.

2.1.3.2 Deploying Topology Generator

Deploy '`server/topologygenerator.war`' to your servlet engine[3]. Make sure that Topology Generator servlet can read its configuration file '`topology_generator.conf`'. Read and write permissions are needed for topology and data files in '`data`' and '`web/topologyPics`' directories. It is probably easiest to grant read and write permissions for whole '`verkkoke`' directory and subdirectories. This should not be a problem.

Execute permission is needed for `dot`. Topology Generator uses reflection, so `java.lang.reflect.ReflectPermission "suppressAccessChecks"` must be set.

Finally, Topology Generator's URL must be updated to '`phpinc/config.php`', either by manually editing the file or by Graphical Configuration Interface. GCI is located in '`web/configGUI`' and should be accessible using web browser and working if web server is correctly configured. Name of the parameter is `$topologyGeneratorURL`. Default value for URL is to assume that servlet engine is run on a local machine (localhost) at port 8180[4].

2.1.3.3 Deploying SCORM Package

Before SCORM package can be deployed to LMS, it must be updated. In '`scorm/simassignment/launch.htm`' find parameter `onLoad="launchSim('http://host/directory/simassignment.php')"`. Edit the URL according to your web server settings to match '`simassignment.php`' in '`verkkoke/web`' directory. Then package *contents* of the '`scorm/simassignment`' directory to a zip file. This zip file is now ready to be deployed to your LMS.

---

[1] All of the mentioned programs are available via apt in Debian GNU/Linux, including Sun's Java Virtual Machine

[2] In Apache, it is easy to use `Alias` and `Directory` commands to publish just about any directory in arbitrary URL (within domain, of course)

[3] In Apache Tomcat this is usually simply done by copying .war file to '`webapps`' directory

[4] This is Debian's default setting

2.1.3.4 Updating Configuration Files

Usually, configuration files should be updated using Graphical Configuration Interface. GCI is located in '`web/configGUI`' and should be accessible using web browser and working if web server is correctly configured. To update configuration files, simply log in to GCI and save configuration once. CGI has explanations of different parameters.

Main Verkkoke configuration file is '`phpinc/config.php`'. Simulation Server and Topology Generator have their own configuration files ('`server/bin/simulationserver.properties`' and '`topology_generator.conf`' respectively), but they are just subsets of parameters in '`config.php`'. If files are edited manually, they must be kept in sync to avoid problems.

2.1.3.5 Starting Simulation Server

Simulation Server is located in directory '`server/bin`' along with '`run.sh`' script to run it. It should be ready to run after configuration files have been updated. Server is able to follow changes made to configuration file and automatically adopts new settings.

2.2 Configuring Verkkoke

The post-installation configuration of Verkkoke 1.1 should be done using web based Graphical Configuration Interface. GCI main script file is '`web/configGUI/index.php`' which should be accessible if your web server is configured correctly. GCI has descriptions of different parameters and other help. Please remember that GCI must be run at least once and configuration saved to synchronize different configuration files in the system.

If you know, or wish to know, what you are doing, you can edit configuration file '`phpinc/config.php`' by hand. File is quite well documented, so it should be relatively easy to change some settings through there, if needed.

There is also a file '`phpinc/resources.php`' that contains resource links that are shown to students if their score is less than 100 %. Edit it, if you wish to add or remove resources.

3 Using Verkkoke as Student

As mentioned in Chapter 1 [Introduction], page 1, Verkkoke 1.1 is designed for use in conjunction with a programming assignment.

To do the programming assignment, the student takes the following steps:

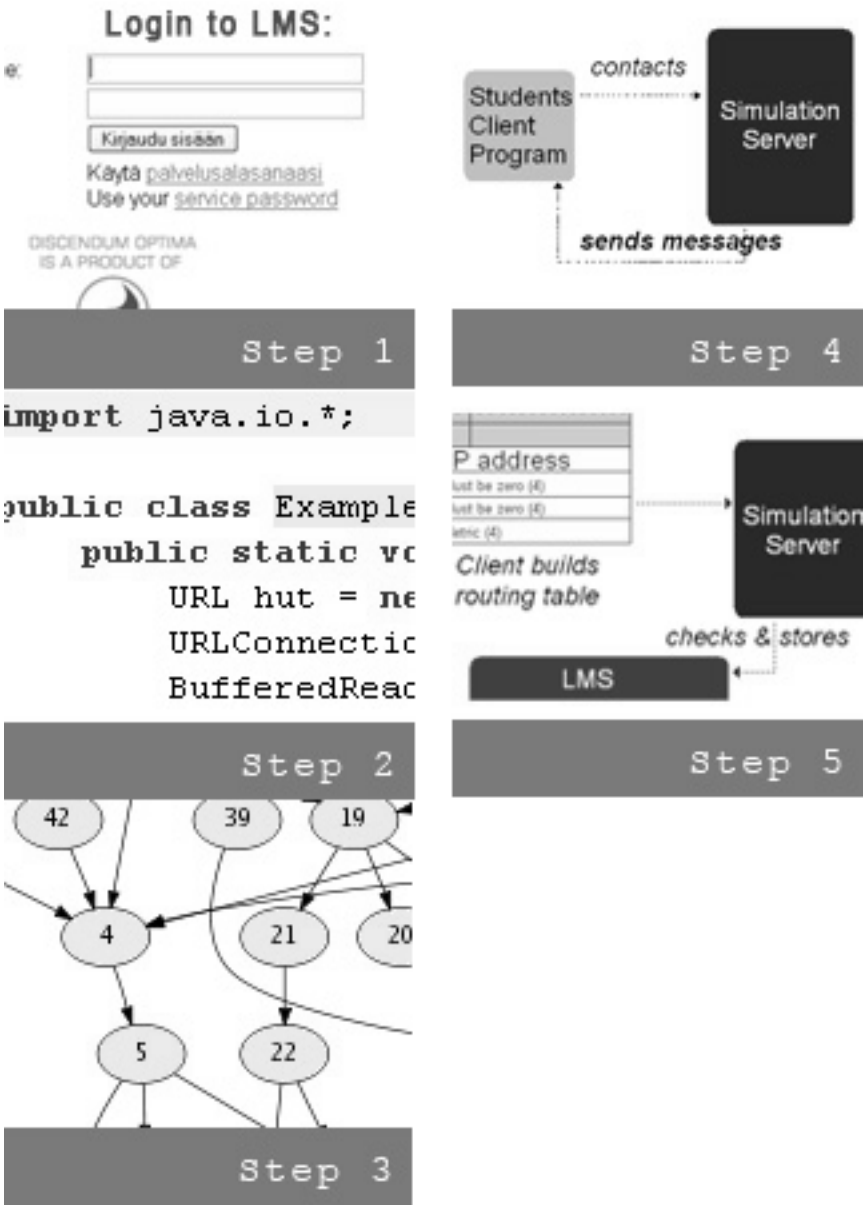
1. The student logs in to the Learning Management System (LMS) to read assignment instructions, to commence the assignment and retrieve assignment results.

2. Externally, the student makes the client program, then logs back into the LMS to start the assignment.

3. The system generates a random topology for the student.

4. The student has the client program contact the simulation server. The simulation server sends routing messages to the client based on the topology generated for the student.

5. The client builds its routing table based on the routing messages and sends that routing table to the server. The simulation server checks whether the routing table is correct and stores the result in the LMS. The student goes back to step 1, to see the result. If the routing table is not correct, the student is allowed to submit again.



Step 3 in the above process description can be reached in two different modes, practice mode and submission mode. When in practice mode, the student can choose the size of the generated topology. A picture of the topology is shown when it has been generated. The student has the option to generate a new topology, if desired. When in submission mode, the system will always generate a new topology of certain, teacher configurable number of nodes, and the topology is not shown to the student.

The student's client builds its routing table based on routing messages sent by the server. The format and content of these messages depend on the simulated routing protocol. Verkkoke 1.1 simulates two routing protocols which are abstractions of real routing protocols. The abstractions are called DV (which stands for Distance Vector) and LS (which stands for Link State), and they are loosely based on the real protocols RIP and OSPF, respectively.

The student's client sends its routing table to the server using a separate submission protocol which uses TCP. The server then compares the submitted routing table with the routing table of the student's node in the simulation.

3.1 The DV Protocol

The DV protocol uses UDP so that the student gets experience with datagram sockets. Because UDP does not guarantee reliable data transfer, and because in order to build its routing table, the client must receive all DV messages, the DV protocol includes a basic reliability mechanism, called stop-and-wait Automatic Repeat reQuest (ARQ). Stop-and-wait ARQ provides reliability by requiring that each packet sent is acknowledged. The sender sends a packet and waits for an acknowledgement (ACK) packet, before continuing. Since packets, also ACK packets, can be lost or duplicated, sequence numbers are used to determine which ACK acknowledges what. In fact, when a receiver acknowledges reception of a packet, it attaches to the ACK the sequence number of the next packet it expects.

First, the student's client sends the authentication message. The server responds with the message `AUTH_OK s`, where `s` is a random integer sequence number from within the range $[1..2^{30}]$. All messages sent by the server contain a sequence number, and the client must respond to every message received with an acknowledgement of the form `ACK s+1`.

Upon reception of the acknowledgement of the `AUTH_OK s` message, the server starts sending the routing data. The routing data messages have the format `[x, y, z, s]`, where `x` is the network interface number, `y` is the name of a router, `z` is the distance to router `y`, and `s` is the sequence number. The sequence number is incremented by one for each new message. The server resends, by default, with one second intervals, if a message is not acknowledged. If no ACK is received in 15 seconds, the server gives up, and stops resending the message. These timeouts can be configured in a configuration file, as discussed in Section Section 2.2 [Configuring Verkkoke], page 4.

Each of these messages are sent in separate UDP packets. In the DV protocol, the distance between two adjacent routers is assumed to be always one. As the client receives the routing messages, it calculates its routing table using a distributed Bellman-Ford algorithm.

3.2 The LS Protocol

The LS protocol uses TCP so that the students get experience with byte stream sockets. The new line character separates different messages. Since TCP provides a reliable byte stream, no additional reliability mechanism is required.

Like with the DV protocol, the student's client first sends an authentication message that identifies the student. If authentication succeeds, the server responds with the message `AUTH_OK`. Upon successful authentication, the client sends a `HELLO` message to the server. In a real routing protocol, the `HELLO` message would be sent through all network interfaces, and thus to all neighbors. Since the simulation server represents the entire virtual network, the client sends the `HELLO` message only to the simulation server. The server responds with a message for each neighbor of the student's router in the simulated network. The response message has the form `HELLO, name, interface`, where `name` is the name of the neighbor that would have sent this response, and `interface` is the number of the interface through which the message would have arrived, e.g. `HELLO, 3, 2`. When responses for all neighbors have been sent, the simulation server sends the message `NOMORENEIGHBORS, 0`.

Next, the client determines the distance to each of its neighbors by sending an `ECHO` message. The server responds with a message for each neighbor of the form `ECHOREPLY, name, distance`, where `name` is the name of the neighbor that would have sent this response, and `distance` is the distance to that neighbor, e.g. `ECHOREPLY, 3, 2`.

Now the client has all relevant information about its neighbors. To be able to build its routing table, the client needs information about the rest of the network. In the LS protocol, routers communicate information about their links with `INFORM` messages. These have the form `INFORM, router_name, neighbor_name, distance`, where `router_name` is the name of the router which is sending its link information, and `neighbor_name` is the name of one of its neighbors, e.g. `INFORM, 3, 5, 2`. `distance` is the distance between the two routers. To get link information from the server, the client sends the message `INFORM`. The server responds by sending `INFORM, router_name, neighbor_name, distance` messages to the client. After all these messages has been sent, the server closes the TCP connection. At this point, the student's client will have enough information to build a link state database and calculate its routing table using Dijkstra's shortest path algorithm.

If the server receives an invalid protocol message from a client, it responds by sending the message `ERROR`.

3.3 Routing Table Submission

The student's client sends its routing table to the server using a separate submission protocol which uses TCP. The server then compares the submitted routing table with the routing table of the student's node in the simulation.

The submission protocol advances as follows:

First, the client sends the authentication message. The server responds with the message `AUTH_OK`. Then, the client sends its routing table in this format: `x,y,z;x,y,z;...;x,y,z;`, where `x` is the name of a destination (router), `y` is the distance to that destination, and `z` is the number of the interface, through which the destination is reached. Each routing table row ends with a semicolon (`;`). The server responds with a message of the form `SUBMISSION OK, feedback`, or `SUBMISSION FAILED, reason`, if the routing table message was invalid or there was some internal error. Finally, the server closes the TCP connection.

The requirements for the submitted routing table are different for the DV protocol and the LS protocol. The difference is that for the LS protocol, if there are multiple shortest paths (of equal cost) with different next hops (and thus different interface ids) for a destination, the routing table must contain an entry for each of these options. This is called Equal Cost Multipath (ECMP) routing. ECMP is required so that the student can demonstrate that the algorithm used in the client finds all the shortest paths. In real-world routing, ECMP can be used for load balancing. ECMP is supported by OSPF. ECMP is not required with the DV protocol, however. The ECMP requirement in the LS protocol also serves to balance the difficulty of implementing the two protocols, since the DV protocol involves implementing stop-and-wait ARQ, whereas the LS protocol just uses TCP.

The submission server compares the submitted routing table with the routing table of the student's node in the simulation. The simulation table contains all ECMP entries. The feedback to the student consists of a score (passed/failed) and the number of incorrect or missing routing table destinations. The submission server sends this information to the student's client. The result can later be accessed through the LMS, on the result page.

4 Using Verkkoke as Teacher

4.1 Learning Management System

A Learning Management System (LMS) is used with Verkkoke 1.1 for managing student assignment statuses and results, submitting documents related to the assignment, e.g. assignment plans, for reporting purposes and for other value-adding functionality that a particular LMS may provide.

Since a university giving a networking course is likely to already have an LMS in use, it is usually desired to use this existing system to manage course information, such as students and their assignment scores.

Typically, a teacher performs the following tasks with Verkkoke 1.1:

- Add students to LMS.
- Accept assignment related documents submitted by students, e.g. plans or final reports.
- Look at the assignment report with the grades for the different assignment parts for each student.

Since the programming assignment grades are automatically stored in the LMS, the teacher does not need to learn how to use any new system. All of the teacher's tasks are done through the LMS, which the teacher is probably already familiar with.

4.2 Backing Up Verkkoke

Normally all result data is stored to the Learning Management System and there should be no reason to store any other data (topologies etc.). If for some reason all data has to be backed up, it is enough to copy directory '`data/student_data`'. If one wishes to backup configuration, file '`phpinc/config.php`' should be copied.

5 Compiling From Source (Optional)

Verkkoke can be compiled using Apache Ant, the build file is "build.xml". For details, refer to http://ant.apache.org/.

Change to the '`server`' directory and issue any of the following commands:

`ant` Compiles Verkkoke.

`ant build_the_war`
: Builds the topology generator WAR (Web ARchive) file.

`ant javadoc`
: Creates the Javadoc documentation. Javadocs are not complete and therefore not very useful

`ant clean` Removes the compiled classes.