Teaching Network Protocol Concepts in an Open-Source Simulation Environment

J. Mark Pullen
Department of Computer Science
George Mason University
4400 University Drive, Fairfax VA 22030
USA
mpullen@cs.gmu.edu

ABSTRACT

This paper describes a new-generation open source tool for computer networking education called the Java Network Workbench 2 (JNW2), along with the successful teaching practices associated with that tool. JNW2 applies the constructivist educational philosophy, where students learn by doing, in this case by creating a software solution to an abstracted problem. The packaging of JNW2 is therefore categorically different from production network simulators such as OPNET and ns3. The paper describes the philosophy and software design of JNW2 and relates these to the educational theory of constructivism. Experience and objective outcomes relating to teaching with JNW2 are described. Students are enthusiastic about using it and analysis shows a clear improvement in exam performance for concepts reinforced via JNW2 exercises.

CCS CONCEPTS

• Networks→Network protocol design

KEYWORDS

Network protocols; teaching computer networking; simulation.

ACM Reference format:

J. Mark Pullen. 2018. Teaching Network Protocol Concepts in an Open-Source Simulation Environment. In *Proceedings of 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '18)*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3197091.3197137

1 INTRODUCTION

This paper describes a new-generation tool for computer networking education called the Java Network Workbench 2. It includes successful practices associated with that tool, in light of the educational philosophy called constructivism.

JNW2 is the third in a family of network simulators intended primarily for educational use. The earlier members of the family are Network Workbench (NW) [6] and Java Network Workbench. This paper builds on concepts originally introduced by the author in [7] and updates those results based on the latest JNW2 software version and teaching experience. Section 2 explains the philosophy behind this approach to teaching; section 3 describes the structure of JNW2; section 4 explains how JNW2 implements a Constructionist approach; section 5 provides quantitative evidence supporting the approach described, and section 6 concludes the paper.

2 PHILOSOPHY BEHIND NW

NW is a software simulation environment for academic investigation of network protocol concepts. The motivation behind NW and its programming exercises [8] began with the observation that students understand network protocols much better after they create a mental model of the protocol workings in the process of programming them. The original NW consists of a collection of software modules written in the C++ programming language, implementing a simplified but complete protocol stack modeled after the primary protocols of the Internet. A discrete event simulation (DES) system provides a means for the protocols to be executed in an observable, repeatable way by students. The students are able to create their own versions of each of the protocols in the stack and test their implementations either as individual protocols or in an integrated stack. NW has been described in detail by the author in [6, 7].

2.1 JNW2 System Overview

The specific context provided by NW is intended to support students puzzling out solutions and do so in such a way that little of their time is devoted to issues that are not central to how the protocol works. For example, the interfaces between stack layers are provided by the Workbench, as are the mechanics of the discrete event simulation. This approach was based on practical observation, made over years of teaching networking, that classroom instruction was found to be less than satisfactory for imparting a real understanding of the protocols' workings, while programming exercises that apply the protocols teach how to use them, not how they work. NW was developed to address the need for hands-on work that teaches how network protocols function. Student responses and examination results indicate that NW works very well for this purpose.

Over the time since NW was introduced, emphasis in Computer Science education has moved away from the C++ language to the Java language. Java features lower complexity, an elegant overall design well-adapted to modern programming practices, and a virtual machine implementation that makes it extremely portable. Initial efforts at developing a Java Network Workbench unfortunately did not reflect the elegance of design and implementation that is possible in Java, so we started again and in JNW2 produced an implementation that we believe is an exemplar of Java software for Computer Science students.

2.2 Computer Network Simulation as a Learning Environment

It is well known that students learn best those concepts that are reinforced by activities requiring them to use the concepts. However, computer networking is a complex, interdisciplinary subject. It is important to avoid assignments that require long periods of time learning details such as socket-level programming that may be useful programming skills but do not teach critical aspects of network protocols. Without an understanding of the internal operation of the network, students will see it as a "black box," learning too little to make effective use of networks in the distributed systems that are common today. A further complication occurs because an operational network environment features constantly changing traffic, where programming errors may not be triggered consistently.

Computer Science students need to understand the most important collection of protocols in use today: the core protocols of the Internet. There is great value in the interoperability made possible by these protocols. However, achieving this interoperability requires adherence to specific rules for information transfer. It is important that students learn how interoperability is facilitated by working with details of a protocol stack that conforms to the protocol definitions. In the Network Workbench, they do this by implementing key aspects of the protocols. This requires both solving a technical problem (programming a protocol that will communicate with another instance of itself) and adhering to a techno-social compact (programming a protocol such that it will interoperate with a different implementation).

The author's experience indicates that a network simulation environment represents the best compromise between the problems associated with student network programming and the need for students to reinforce classroom learning by doing a real project. Simulation is used widely to abstract away non-essential physical details (which also are likely to be unrepeatable in testing) while exercising the critical aspects of a protocol. Simulations run as user programs in the computer, making them easy to load and run in a personal computer. Students can program abstracted versions of the protocols that actually run in the simulation environment. The simulation environment is structured as high-quality Java code, designed to facilitate observation of data flowing through each level of the protocol stack. Students also are exposed to the concept of network simulation, an important tool in protocol development and network design.

Table 1. JNW2 Java Packages and Classes

(* indicates multiple similar files)

Source Package	**Purpose**	**Constituent Classes**
JNW2	Top-level functionality	ConnectivityMatrix, Constants, Copyright, EmailApplication, GenericTest, RunSimulation, SimLogger, SimualtionEngine, Topology
JNW2.config	Network configurations for exercises	ConfigDescription, Dijkstra, LANCollisions, ReliableDLC, ReliableTransport, Run123, SimpleWAN, SlowStartWindow, WAN2
JNW2.data	Test input	Email*.txt, Stream*.txt
JNW2.events	Discrete Event	DiscreteEventSimulator, DLCSendWakeupEvent,

	Simulation subsystem	LANSendRecycleEvent, PhysicalReceiveEvent, ReceiveEvent, SendEvent, TLSendWakeupEvent, TransportReceiveEvent, TransportSendEvent
JNW2.gui	Network drawing subsystem	CollisionUtil, ConsoleOutputer, FileManager, GraphicMouseAdapter, GraphicPanel, Gui*, Jnw2GuiFrame, MenuLink, OtherInputPanel, RoutingMatrixCellEditor, RoutingMatrixDialog, RoutingMatrixTableModel, State, StringOutputer, SubnetDetailsDialog, TopologyBuilder
JNW2.stack	Protocol stack modules	ApplicationLayer, DlcLayer, Layer, NetworkLayer, PhysicalLayer, Stack, TransportLayer
JNW2.utility	Miscellaneous processes	DataUtility, DijkstraRouting, MessageGenerator, RandomNumber, RandomNumber2, Statistics

3 STRUCTURE OF JNW2

3.1 Components

Very powerful network simulators are available today. However, those which work at the protocol level, such as OPNET [3] and ns3 [4] are highly complex tools that take a long time to master. JNW2 was designed to maintain the simplest structure consistent with valid representation of the TCP/IP protocol stack, while also providing scaffolding for aspects of simulation and module interface functionality which does not contribute to student understanding of protocol operation. JNW2 was developed by the author of this paper with support from Computer Science graduate students who were at the time employed as professional software developers. A collection of assignments, described below, is distributed with the Java project containing all JNW2 code except exercise solutions. The student is able to see and, if desired, modify all of the code. The Java packages and classes comprising JNW2 are summarized in Table 1 in order to give educators who may want to adopt it an understanding of the protocols it covers.

3.1 Basic Mechanisms Embodied in JNW2

Internetting: JNW2 models an abstracted version of the Internet architecture. Each node in the simulated network has two integer attributes: *networkNumber* represents the element in an Internet address that is used to deliver packets to the physical network. All nodes on the same physical network have the same value of *networkNumber*. The other attribute, *hostNumber*, is equivalent to the host number in an Internet address. In addition to these two, each interface of each node has a globally unique *ifaceNumber* that identifies the node's physical address on a serial link or LAN subnet. Other values of *ifaceNumber* connect to serial links among nodes. The *nodeNumber* one on each LAN is assigned to the associated network layer router.

Discrete Event Simulation: In a computer network, many asynchronous (unsynchronized) events occur constantly, yet the whole network must function as a coordinated, distributed system. Discrete Event Simulation (DES) is widely used to study complex systems with such properties, by abstracting for study the key distributed system functions [5]. In JNW2, every simulated event is discretized to a time that is represented by a long integer value *simulationTimeInTicks*, a count of some basic time unit such as microseconds. Every event happens on a specific value of *simulationTimeInTicks*. The DES routines are responsible for accounting, in an efficient way, for all events that "have not yet happened." Typically, the result of an event happening is that one or more new events are scheduled. When events "happen" they invoke Java functions. This arrangement provides a simple mechanism for representing the multiple, asynchronous threads of control needed to simulate a network.

JNW2 DES Function *nextEvent( )* causes the function for the next event "waiting to happen" to be invoked. The top level of the JNW2 program invokes this function indefinitely, until the event list is empty or a time limit is reached. The DES routines maintain a two-dimensional linked list of events, with the first dimension ordered by "happen time" and second dimension FIFO by order of scheduling within a particular "happen time." This arrangement supports efficient event list search and update.

Stochastic simulation: JNW2 uses pseudo-random number generators to create two types of events critical to behavior of networks: message arrivals and datalink errors. Each message stream and each datalink has its own independent stream of pseudo-random events, produced by random number generator functions. The built-in functions are deterministic (fixed time interval) for best-effort and reliable messages, which are intended to reflect the behavior of humans generating email. The Poisson distribution (exponential inter-arrival) is used for bit errors. JNW2 contains its own random number generators based on the mixed congruential method [2]. The pseudo-random number sequences are seeded individually with values that are the same from run to run, so that the events, while they have "random" statistics, always occur at the same point (a considerable help in debugging). To change a random number generator distribution, the student needs only to provide a generator function.

Input/Output: As scaffolding for exercises, files containing a collection of "email" inputs, the WAN topology, and the number of hosts on each LAN are included in the JNW2 Java project. The files use a simple text format that can be edited by students. At the end of simulation, summaries of the message traffic at all nodes are automatically displayed as output. A graphic network drawing package to create the network description files is included. Figure 1 shows the network drawing interface.

3.3 JNW2 Exercises

The JNW2 download includes assignments and stub modules coordinated with the project book *Understanding Internet Protocols* [8], which support exercises. The book includes cross-references to popular networking textbooks such as [10] and [11], showing where the exercises can be combined with textbook material. This includes studies where the student programs:

- *bit stuffing/unstuffing* for frame formatting
- *error detection* via cyclic redundancy check calculations
- *Data Link Control* (DLC) flow and error control, in which the student programs the decision logic for Automatic Request for Retransmission.
- *Carrier Sense Multiple Access with Collision Detection* (CSMA/CD) binary exponential backoff as used in Ethernet for resolution of collisions
- *topology matrices* hand coded to describe the WAN and use of graphic editor to produce such matrices
- *network layer routing* functions for route optimization and packet forwarding
- *reliable transport layer* function that implements the core logic of the sending end in a protocol abstracted from the Transmission Control Protocol (TCP)
- *slow-start sending end window* abstracted from the approach used by TCP for congestion control

Additional exercises from [8] that are planned for addition to JNW2 are:

- *token passing* local area networks dealing with a received token
- *network layer routing information distribution*, in which the student programs the function that updates Link State Advertisements
- *multicast networking* function that determines the set of WAN interfaces participating in the multicast tree.
- *network security* filter function for a firewall
- *application layer message handling* function for an email list server.
- *internetworking* where all previous project parts are combined with the result that reliable transport of email messages is interspersed with multicast stream transmission.

4 JNW2 IMPLEMENTS CONSTRUCTIVISM

The motivation behind JNW2 and its exercises began with the observation that students understand network protocols much better after they create a mental model of the protocol workings in the process of programming them. JNW2 therefore is intended specifically to support that puzzling process in such a way that the student's time largely is devoted to issues that are central to how the protocol works. For example, the interfaces between stack layers are provided by the JNW2, as are the mechanics of the discrete event simulation – students would learn little about protocol operation by building these. The educational philosophy involved here, that most closely aligned with JNW2, is constructivism [1]. The following compares the philosophy and practices associated with JNW2 to the eight constructivist design principles set forth both by educational pioneers Savery and Duffy [9].

1. *Anchor all learning activities to a larger task or problem.* Each JNW2 exercise takes place in the context of a complete working protocol stack, which is in turn presented in the context of understanding how the Internet works.

2. *Support the learner in developing ownership for the overall problem or task.* The student is completely responsible for solving the problem (programming the protocol) while JNW2 provides all supporting software. Supporting documentation online and in [8] provides a collection of useful programming and debugging techniques. These are presented without specific directions so that their employment is up to the student.

3. *Design an authentic task.* The protocols are real, although simplified. Moreover, for a working solution they must function as part of a larger whole, an interoperating protocol stack.

4. *Design the task and the learning environment to reflect the complexity of the environment they should be able to function in at the end of learning.* Each protocol is simplified to the point where it has just the basic properties the student needs to understand at the completion of an introductory course. In an advanced graduate course, the students who use JNW2 design their own protocols at whatever level of complexity is needed for projects which they propose themselves.

5. *Give the learner ownership of the process used to develop the solution.* The student programs the protocols and runs the solutions independently. Coding, debugging and testing are up to the student. JNW2 provides output of solutions for comparison. (Student grades are based on correctness and quality of their code in addition to its output.)

6. *Design the learning environment to support and challenge the learner's thinking.* The JNW2 projects become progressively more difficult and build on what was learned in previous projects.

7. *Encourage testing ideas against alternative views and alternative contexts.* Multiple protocols exist at Datalink and Transport layers for comparison. Also, the student can create a completely new protocol at any layer.

8. *Provide opportunity for and support reflection on both the content learned and the learning process.* The final exercise assembles all of the student's projects into a working whole.

5 GRADE OUTCOMES

At our institution, JNW2 is used in introductory computer networking courses at both undergraduate and graduate levels. The undergraduate course is part of our ABET-accredited program, which requires that an outcomes analysis is performed each time it is offered. We used this to compare outcomes for topics that are reinforced by use of JNW2 exercises versus those topics where JNW2 does not offer reinforcement. Exam questions deal with general concepts in networking, not specifics of JNW2 projects. The author has available data for a total of four such course offerings, presented to a total of 113 students and summarized in Table 2, which contains examination grade averages weighted by question value. The data were obtained by item analysis of mid-term and final examinations supporting our ABET outcomes report. (JNW2 is used in an introductory graduate course also, but ABET outcomes are not reported.) In the five outcomes where learning was reinforced by JNW2 exercises, the students scored almost one letter-grade higher than they did on the four outcomes not reinforced.

Table 2: Summary of JNW2-related ABET Outcomes

Semester	Number of Students	JNW2-related	Not JNW2-related
Fall 2009	26	64.6	61.3
Spring 2014	30	67.7	56.2
Fall 2014	22	78.0	72.0
Spring 2016	35	78.1	63.5
Overall	113	72.2	62.7

6 CONCLUSION

After nearly twenty years of using the approach reported above we have concluded, both from subjective assessment and objective analysis of outcomes, that programming key aspects of computer network protocols strongly reinforces student learning. Furthermore, students often comment in course-end evaluations that they find JNW2 to be a productive and enjoyable way to gain understanding of Internet protocols. Moreover, we are pleased to report that JNW2 and the associated exercises are available as open source to all educational programs. The current project zipfile is available at http://netlab.gmu.edu/JNW2 and solution code will be provided to teaching faculty who send a request to the author under their institution's letterhead.

REFERENCES

[1] Gruender, C. D. Constructivism and learning: A philosophical appraisal, *Educational Technology 36*, 21-29, 1996

[2] Hillier, F. and G. Lieberman, *Operations Research, 2nd Edition*, Holden-Day, 1974, pp 626-628

[3] Katzela, I., *Modeling and Simulating Communications Networks*, Prentice-Hall, 1999

[4] ns-3 Tutorial, http://www.nsnam.org/docs/release/3.14/tutorial/singlehtml/ last visited 14 Jan 2017

[5] MacDougall, M., *Simulating Computer Systems: Techniques and Tools*, Chapter 1, MIT Press, 1987

[6] Pullen, J.M., The Network Workbench: Network Simulation Software for Academic Investigation of Internet Concepts, *Computer Networks Vol 32 No 3*, 365-378, March 2000

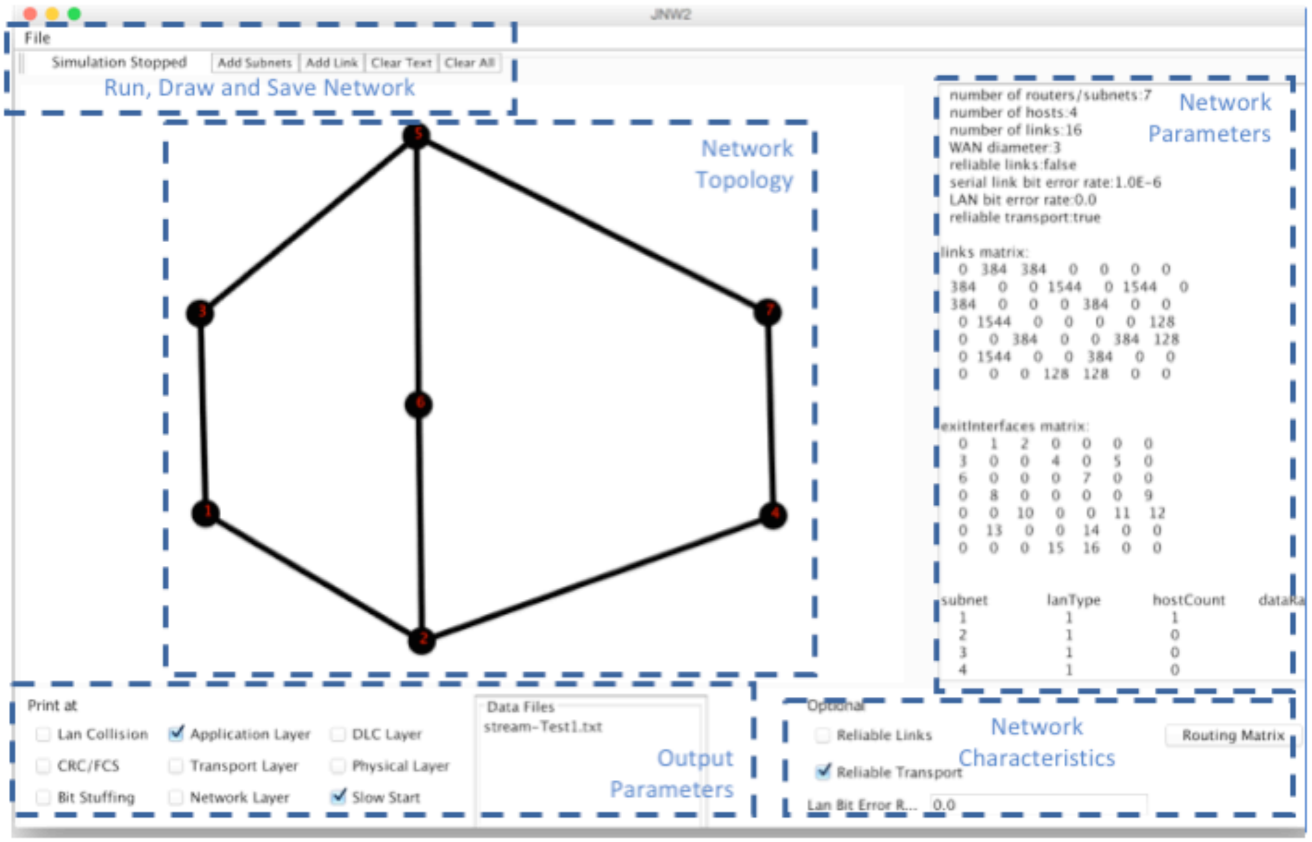
[7] Pullen, J.M., The Network Workbench and Constructivism: Learning Protocols by Programming, *Computer Science Education Vol 11 No 1*, 1-14, September 2001

[8] Pullen, J., *Understanding Internet Protocols Through Hands-on Programming*, John Wiley & Sons, January 2000 (out of print but available at http://netlab.gmu.edu)

[9] Savery, J. R. and T. M. Duffy, Problem based learning: An instructional model and its constructivist framework, *Educational Technology 35*, 31-38, 1995

[10] Stallings, W., *Data and Computer Communications (10th Ed.)*, Pearson, 2014

[11] Tanenbaum, A. and Wetherall, D., *Computer Networks (5th Ed.)*, Pearson, 2011



```
number of routers/subnets:7
number of hosts:4
number of links:16
WAN diameter:3
reliable links:false
serial link bit error rate:1.0E-6
LAN bit error rate:0.0
reliable transport:true

links matrix:
  0  384  384    0    0    0    0
384    0    0 1544    0 1544    0
384    0    0    0  384    0    0
  0 1544    0    0    0    0  128
  0    0  384    0    0  384  128
  0 1544    0    0  384    0    0
  0    0    0  128  128    0    0

exitInterfaces matrix:
  0   1   2   0   0   0   0
  3   0   0   4   0   5   0
  6   0   0   0   7   0   0
  0   8   0   0   0   0   9
  0   0  10   0   0  11  12
  0  13   0   0  14   0   0
  0   0   0  15  16   0   0

subnet     lanType     hostCount     dataRa
  1           1            1
  2           1            0
  3           1            0
  4           1            0
```

Print at

- [ ] Lan Collision
- [✓] Application Layer
- [ ] DLC Layer
- [ ] CRC/FCS
- [ ] Transport Layer
- [ ] Physical Layer
- [ ] Bit Stuffing
- [ ] Network Layer
- [✓] Slow Start

Data Files
stream-Test1.txt

Optional

- [ ] Reliable Links
- [✓] Reliable Transport

Lan Bit Error R... 0.0

Figure 1. JNW2 Network Drawing GUI, Annotated with Descriptive Overlay