# Collaborative Proposal
# TinkerNet: A Low-Cost Networking Laboratory System

Mike Erlinger, Mart Molle

June 17, 2004

## 1 Introduction and Motivation

The 2002 SIGCOMM Workshop on Educational Challenges for Computer Networking [11] exposed many issues related to teaching computer networking: top-down versus bottom-up approach; one course versus many courses; required course versus elective course; and undergraduate versus graduate emphasis.

Throughout the discussions one recurring theme emerged: the need for a laboratory to augment lecture. While the principles of networking can be presented in lectures, the group recognized that real understanding occurs when students actively develop and evaluate systems based on those principles – there is no good substitute for hands-on experience with real networks [9] [10]. Many different approaches to networking laboratories were discussed including vendor-specific laboratories, exercises on operational networks, and stand-alone laboratory environments. All of the discussed laboratories share a few common issues: initial cost of the laboratory and cost of continued maintenance. Our (Harvey Mudd College and University of California, Riverside) response, TinkerNet, is a low-cost, flexible, stand-alone laboratory for running networking experiments, which combines ideas from various papers [14] [13] [2] [17]; with open source software [6] [7] [16]. TinkerNet differs from most of the laboratory environments presented at SIGCOMM in that TinkerNet uses well-known open-source software and inexpensive "obsolete" hardware, representing what we believe is a novel and powerful paradigm for teaching undergraduate networking.

Our combined objectives are to create, use, evaluate, and distribute TinkerNet. This proposal seeks support for the creation of a documented, tested, assessed, and evaluated TinkerNet suitable for distribution to the community. To accomplish this, we need to: document the required hardware, bulletproof the software (an alpha version has been developed), elaborate the set of experiments (some already prototyped), and develop a framework so that the user community can continue adding new experiments.

## 2 TinkerNet

TinkerNet (Figure 1) is a stand-alone laboratory (initially designed for teaching networking, but surely adaptable to Operating Systems and other courses) consisting of an array of back-end nodes, each containing two Ethernet interfaces. One of these interfaces is attached to the *test network* (e.g., 192.168.100/24); the other interface is connected to the *admin network* (e.g., 192.168.200/24). The *admin network* provides

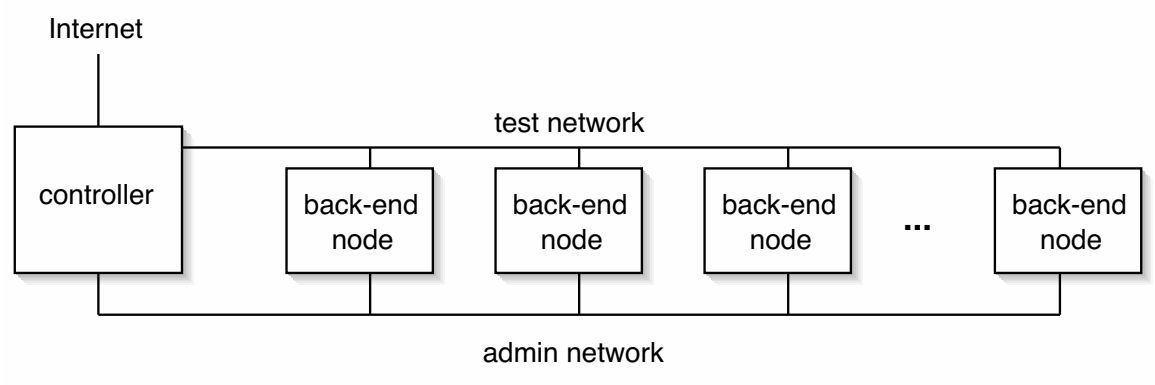


Figure 1: TinkerNet Architecture

each node with both connectivity and administrative services (sending kernel images, remote control of the bootloader, etc). The *test network* is what the students use to evaluate their network kernels. An important security feature of TinkerNet is that neither the *admin network* nor the *test network* is connected to the institution's production network and/or the public Internet.

Students load their OS kernel on a back-end node. These nodes have negligible hardware requirements: a CPU, two network interfaces, minimal memory, and network card eproms for our modified bootloader. In our prototypes the back-end nodes are all Pentium 200s cast off by other departments as obsolete. No monitors are necessary, as each back-end node is accessed solely by its network interfaces.

The TinkerNet controller connects to both the *test network* and the *admin network*, as well as to the institution's normal network. The controller provides a home for the students and for all the software to make TinkerNet operational.

## 2.1 TinkerNet Operational Overview

When using TinkerNet, students are provided with a skeleton source tree containing the function prototypes they must implement and a makefile which will compile their code into a kernel for a back-end node. Having compiled their code, the controller allows the students to remotely boot their kernel and view its output. The student need not be aware of the existence of the *admin network* and its infrastructure. When the student is done testing a particular build of their kernel, they can simply push a button on the controller's GUI and have their node rejoin the pool of available ones.

TinkerNet is focused on the Build-Your-Own-Stack (BYOS) activity common in many networking courses. TinkerNet allows students the opportunity to experience networking protocols by having direct access to an Ethernet network's raw frame data. The experiments are focused on students building and testing pieces of the network protocol stack, from OSI layer 2 (Link) through layer 7 (Application).

### 2.2 TinkerNet Implementation Overview

TinkerNet is based on commodity hardware and the readily-available OSKit[6][7], Linux, and GNU software. Our software choices are widespread within the computing community, and thus TinkerNet will both benefit from the symbiotic relationship with these other projects and have more acceptance because it involves well known and easily available technology. An additional advantage of the TinkerNet approach is that it will be accessible even to those institutions (e.g., undergraduate institutions) that do not have on-going research in the area of computer networks.

Student kernels are built using OSKit (students need no knowledge of OSKit, as everything is pre-configured). OSKit makes prototyping and development of operating systems and OS kernels easier by providing a set of component libraries that can be added or removed from the kernel as necessary. TinkerNet does not need most of the features common to fully functional operating systems (file-system, TCP/IP stack, etc), so most of the OSKit provided modules are not utilized. However, the existence of these modules means that TinkerNet is easily expandable to topics beyond networking, like Operating System fundamentals. Part of the effort of this proposal is to document and organize TinkerNet's use of particular OSKit modules and to identify where expansion into other computing subjects can easily take place.

In prototyping TinkerNet we found the need for some custom modules: a simple scheduler and network stack, an ARP table, initialization functions, and a network based *printf*. All of these have been prototyped and are functional. A main focus of this proposal is to bulletproof and document these additional modules.

## 3 Laboratory Experiments

In developing our prototype we created sketches of a semester-long set of laboratory experiments focused on student development of a fully functional network protocol stack. A sample set of experiments is presented below and a full description of an experiment is presented in Appendix A (a major activity of the proposal is to formalize sets of experiments). In this sample set each new experiment builds on the previous experiments. After a brief introduction, we progress from raw Ethernet packets to fully functional IP and then to UDP. Later assignments build on this by having students create custom protocols over UDP. Many other experiments could be created, e.g., implementing a subset of TCP which recovers dropped packets (a full implementation of TCP would not fit in a semester). We envision TinkerNet being used in advanced courses to implement application protocols and/or network devices like a router.

### 3.1 Example Laboratory Experiments and Goals

- Lab 1: gain proficiency with C (not C++) programming; [1] complete basic networking exercises.
- Lab 2: gain familiarity with the lab environment; successfully compile a TinkerNet kernel; implement functions that send and receive Ethernet packets.

[1] In the standard undergraduate curriculum at both UCR and HMC, the majority of the programming is in C++ and many students will be unfamiliar with the differences between C and C++.

- Lab 3: gain more familiarity manipulating TinkerNet kernels; implement functions that send and receive ARP packets.
- Lab 4: implement a simple, end-host version of IP, the Internet Protocol.
- Lab 5: implement and exercise UDP, the User Datagram Protocol.
- Lab 6: design, create, and implement a peer-to-peer protocol. Use this protocol to locate other hosts on the network running the same protocol.

# 4 Implementation Plan and Deliverables

Based on our experience in creating the TinkerNet prototypes and using TinkerNet in courses at UCR and HMC, we believe that in one year we could develop TinkerNet into a fully-functional mature laboratory environment including a complete set of exercises. During this one year period we would:

1. Create a design document of the TinkerNet architecture: hardware, software, overall operation and use.
2. Document the hardware infrastructure. In creating the UCR TinkerNet prototype, we realized that detailed documentation is necessary for quick assembly of a new TinkerNet.
3. Debug and document the software environment (the prototype software is still in the alpha phase). We need to organize the OSKit modules and refine our additions (administrative network stack, ARP table, scheduler, and *netprintf*). Again, creating the UCR TinkerNet uncovered numerous software issues.
4. Create a set of laboratory experiments, including evaluation procedures.
5. Assessment of student learning in the TinkerNet environment. We will document our current process and search out other assessment procedures.
6. Present TinkerNet at SIGCOMM and SIGCSE. We will present at both these venues to demonstrate TinkerNet, discuss assessment of student learning, and seek community input. There has already been one paper on TinkerNet [15] and a poster session, but neither stressed pedagogical evaluation.

# 5 Equipment and Instrumentation

Because TinkerNet is primarily throw-away equipment, we are seeking minimal support for equipment. Rather our request focuses on support to develop a stable TinkerNet environment.

### 5.1 HMC Development Environment

**Development Staff** – The students who prototyped TinkerNet are the type of students who would create the full TinkerNet environment. While these are undergraduates, they have significant experience in developing and maintaining computer systems. This experience comes in part from two HMC development experiences: *Student Staff* and *Clinic*. At HMC the Computer Science department employs a *Student Staff* (undergraduates) to administer the department's computing network. Our *Clinic* program is a year long project proposed and supported by an industrial client which exposes our students to large software development projects. For example a recent Clinic created and implemented a security related network protocol (soon to be an Internet standard). Thus, our undergraduates provide a pool of students capable of designing, implementing, and caring for TinkerNet.

**College Resources** – Existing HMC college computing facilities will be more than adequate for our development needs. In addition to the personnel funded under the proposal, HMC has committed to supporting at least one additional undergraduate programmer for each summer. The College will also partially support the involvement of its assessment office in TinkerNet evaluation.

### 5.2 UCR Development Environment

**Development Staff** – We anticipate hiring a highly qualified graduate student to support UCR's involvement with the TinkerNet project. This student will be responsible for: creating and maintaining both our physical and virtual TinkerNet clusters; contributing to bulletproofing the TinkerNet environment; and providing training and consulting to the Teaching Assistants assigned to the UCR classes that will be pilot testing TinkerNet.

**College Resources** – As a research university, the facilities at UCR are well equipped to handle an undertaking like this one. The Department's instructional computing facilities are maintained by its own system administrators, which includes three full-time staff members and four part-time students. The Department's system administrator staff will work with the TinkerNet development staff to assist with TinkerNet clusters, and with the diagnosis and solution of any major problems that we encounter during development and pilot testing.

## 6 Experience and Capabilities of Investigators

**Michael A. Erlinger** is Professor and Chair of Computer Science at Harvey Mudd College. He has taught computer networking each year for the last 10 years, having developed the HMC networking course. For 20 years at HMC he managed the departmental computing environment with students as the primary system administrators. His research areas are computer networking and network security. He is currently co-chair of the Intrusion Detection Working Group within the IETF, where a series of protocols are being developed for intrusion detection systems. He brings experience in software development and software distribution from his years of industrial activities. Through the Clinic program he has experience with students developing large software projects outside of the classroom.

**Mart Molle** is a Professor of Computer Science and Engineering at UC Riverside. He has been active in the development and teaching of computer networking courses for over 20 years — first at the University of Toronto, and more recently at UCR. He also has considerable experience with the detailed specifications of network protocols through his contributions to the IEEE 802.3 Ethernet Working Group. He chaired the IEEE 802.3w Enhanced Medium Access Control Protocol Task Force, and was the primary author for the changes to the Ethernet CSMA/CD protocol specification for supporting Gigabit Ethernet. He currently serves as the Technical Reviewer of requests for Ethernet Type Field assignments on behalf of the IEEE Registration Authority.

### 6.1 Results From Prior NSF Support

The PI (Erlinger) has not had any NSF support in this area. The Co-PI (Molle) is a co-investigator in the NRT consortium led by UCLA: *Scalable Research Testbed for Next generation Mobile Wireless Networks* (10/1/2003 - 9/30/2006). The goal of this project is to build cross-layer simulation models and implementation platforms to create a large scale wireless testbed. UCR share of the total budget is $275,000. The work is still at an early stage and no results have yet been published.

## 7 Collaboration

See Appendix B for a discussion of collaboration efforts between Harvey Mudd College and University of California, Riverside.

## 8 Evaluation Plan

As with any academic endeavor, assessment of this project is both vital and difficult. It is not possible to test the same student under two different circumstances, as their additional knowledge gained in the first testing confounds skills assessment. We have developed an initial assessment plan to measure the impact of the proposed laboratory on student learning at HMC and UCR. During the year we will attempt to discern the extent to which TinkerNet increases student understanding of computer networking. This plan was developed in collaboration with Dr. Margaret Kasimatis, HMC's Executive Assistant to the President for Assessment. Dr. Kasimatis will help refine and guide the assessment plan, and aid in analysis of the collected statistics.

### 8.1 Assessment

Using the TinkerNet prototypes we will test students in the first few classes of each course on their understanding of the operation of a protocol stack. During the course specific questions will be asked prior to each experiment. Some of these will be oriented towards information that students should already know from lecture while other questions will pertain to information particular to the experiment.

After each experiment, students will be given a similar set of questions. Hopefully, by comparing the results we will be able to evaluate how well each experiment re-enforces material the students have seen previously and how well material becomes understood by having performed the experiment. We will also continue to monitor the literature for new approaches to assessment of student learning, in particular material related to laboratory experiences.

Since students will be involved in creating TinkerNet, HMC will also develop a set of questions pertaining to the experience of creating and maintaining a large software project. Prior to the project students will be questioned on the issues related to the software development process. During TinkerNet development, students will be retested, trying to discern what they have learned during the creation of a real software system. Finally, we will make our assessment material available to others as part of the TinkerNet package.

# 9 Dissemination of Results

The creation of TinkerNet has potential for broad impact in computer science education. Many educators have expressed a desire to create a computer networking laboratory with well defined experiments. Even before TinkerNet has reached full maturity, we expect to present our experiences in two major venues: the annual SIGCSE conference and the SIGCOMM workshop on networking education. One of the results of the recent SIGCOMM Workshop was to ensure that the workshop becomes a standing part of SIGCOMM.

When TinkerNet has reached a working level of maturity, we intend to publish papers in the usual places, and make the software available to anyone interested. We will build a Web page that includes a CVS tree with design documents, code, laboratory experiments and instructions on building and using TinkerNet. We will then advertise TinkerNet on appropriate mailing lists and at conferences focused on computer science education, particularly computer networking education, e.g., SIGCOMM and SIGCSE. It is our intention to eventually seek NSF National Dissemination support.

# 10 Conclusion

TinkerNet is a low-cost, flexible, stand-alone laboratory for running networking experiments. We have prototyped TinkerNet and believe that a full implementation is readily achievable, especially with the joint efforts of HMC and UCR. An advantage of the TinkerNet approach is that it will be accessible even to those institutions (e.g., small colleges or community colleges) that do not have on-going research in the area of computer networks.