TEACHING PRACTICAL COMPUTER NETWORKING WITH LIMITED RESOURCES

Jacky Mallett, Marcel Kyas, Stephan Schiffel

Dept. of Computer Science, Reykjavik University

ABSTRACT

Computer Networks is an important, and often compulsory, course in most Computer Science curricula. Teaching it is often challenging due to the abstract nature of the subject, and the wide range of material that has to be covered. At the same time, an understanding of core concepts in computer networking is increasingly important to students, due to the increasing proliferation of networked devices, and the associated challenges in designing and developing networked and distributed applications. In this paper we discuss our experiences in bringing a stronger practical content to this course over several years,following the Conceive Design - Implement - Operate (CDIO) philosophy. This introduced a series of carefully designed practical assignments throughout the course, building on traditional simple client-server program exercises, through a puzzle based assignment using hand crafted packets, to the final project which involves the construction of a collaborative peer-to-peer network running on student laptops involving the entire class. We will discuss how the practical content is purposefully designed to support the more theoretical aspects of the course, as well as some of the technical challenges encountered.

KEYWORDS

Computer Networks, Project-Based Learning, Standards: 6, 7, 8

INTRODUCTION

Computer networks is a core subject in the modern computer science curriculum, and represents a significant body of theoretical and practical knowledge which provides an important basis for other advanced subjects like Distributed Systems and Internet of Things. Computer networking's principles are core to a wide range of critical economic and technical systems, ranging from distributed computing, through any form of programming that operates on more than one core or central processing unit (CPU), high performance computing, and the daily operations of core business IT systems. Unfortunately it has long been regarded as a highly technical, complex, dry, and difficult course by its students (Sarkar, 2006). Although it is widely recognised that it is a course that can benefit greatly from providing a practical design and implement experience, the dedicated networking equipment and laboratories to provide this experience are expensive and not always available.

Reykjavik University places an emphasis in its scientific instruction on hands-on learning with an emphasis on practical preparation for the challenges students will face after graduation.

The CDIO standard 6 suggests a physical learning environment that supports a hands-on learning process on a networked system. Computer networking equipment is typically expensive and practical training facilities are often confined to industry or only sufficient for a small number of students. We want to follow the CDIO emphasis on active learning in Standard 8.[1] When teaching the compulsory third year computer networks course, Tölvusamskipti (TSAM), however, adopting this approach had been problematic, as the university did not have the laboratory resources in support this approach, particularly for the large class size (200+ students) taking our compulsory third year computer networking course. Prior to the introduction of the changes described in this paper, practical exercises in the course had been restricted to analysing protocol traces using Wireshark, and re-implementing existing protocols such as the trivial file transfer protocol(TFTP) (Malkin & Harkin, 1998). The course was generally unpopular with its students, routinely described as "ridiculously hard" in student comments, and overall student performance was poor with an unacceptably high failure rate, particularly for a compulsory course.

The steps taken to address these issues that we discuss in this paper were relatively small. The lecture content remained substantially the same, although it was switched from following the Kurose and Ross (2016) approach which introduces the network stack from the application level and proceeds to the physical level at the end of the course, to the bottom up approach used by Tanenbaum and Wetherall (2011) and Dordal (2020) which does the reverse. A slightly larger emphasis on network programming was introduced in order to support the new projects. The project content was completely revised, and a series of projects were introduced that provided a hands on laboratory experience following the CDIO standard for Engineering work spaces in Standard 6, and a final project which emphasised CDIO Guided Design and Implement standard 5, using the student's own laptops as a development platform. The projects culminated in a class wide exercise which required students to create a class wide live peer-to-peer network providing an interactive messaging service for all participants. By dint of using the student's personal laptops as the nodes in this network, with some minimal support from instructor servers running on university facilities, it was possible to create these exercises without requiring any additional resources. Reykjavik University does have the advantage of having a state of the art backbone Cisco network which is a consideration, but the projects described here require little more than basic Internet connectivity, and could be attempted either on campus, or from home networks.

Following these changes, the course has now been taught successfully for four years, two of which were online due to the Covid Pandemic. Student reaction to the changes has been extremely positive, performance on the course has significantly improved, and the failure rate has normalised to typical third year levels. Although the course is still regarded as challenging by students, the project experience, and in particular the collaborative last project in peer-to-peer networking have changed student perception to the course to one where they "learn a lot", and even have fun.

[1]CDIO Standard 3.0 CDIO Standard 3.0

TEACHING COMPUTER NETWORKS

In their comprehensive review of 261 Journal and Conference papers on teaching computer networks, Prvan and OžEGOVIć (2020) distinguish four broad strategies in computer networks teaching: using network simulators, either physical laboratories or virtual simulators such as Network Simulation (NS2/NS3)[2] (Gupta, Ghonge, Thakare, & Jawandhiya, 2013), packet tracing where the detailed structure of networks packets is examined using tools such as Wireshark, visualisation techniques with videos, or interactive environments (Getchell, Miller, & Allison, 2005)[3], and virtualisation where laboratories are provided either by software environments, or remote facilities - an approach often favoured in industry for specialised manufacturer based training. To some extent the approach may depend on whether the focus of the course is the design and configuration of network equipment, or a broader focus on networking as a programming problem. Virtual simulators are supported by all the major manufacturers and can provide an excellent way of learning how to setup and configure networks (Gusev, Ristov, & Donevski, 2014), and this approach is also strongly favoured in industry for training to use network equipment. However although the skills required to setup and administer network equipment are congruent with those required to develop distributed and networked applications, they are not identical. At Reykjavik we provide a programming focus for our general networks course, and offer an optional more specialised network design and configuration course for those who wish to take it.

The typical undergraduate introductory computer networks course has a heavy lecture component, following the bedrock laid down by Tannenbaum's Computer Networks(Tanenbaum & Wetherall, 2011) first published in 1981, and still used widely today. This book was originally organised on the OSI 7 layer model for computer networks, and has been adapted over time to include newer technologies, remove obsolete ones, but retains its original format of introducing the network layers one at a time, beginning with physical connections, and ascending the application stack. Naturally, other textbooks, notably Kurose and Ross (2016) have developed the subject in the other direction, beginning with the application layer, today represented by web servers and the accompanying HTML protocol, and then descending the network stack to end at the physical level. Both approaches have entrenched proponents. The top down approach has the advantage of introducing networking with concepts that students have already encountered, and are familiar with, whilst the approach of beginning with the physical level has the advantage that concepts build on each other more or less naturally, and to some extent follow the historical development of the technology itself from small local area networks to the global network of networks, the Internet.

From the perspective of teaching computer networks using programming exercises, there is an advantage to Tanenbaum's order, in that the detail of network communication using packets, with packet headers and protocol exchanges is arrived at earlier, allowing early student exercises to be constructed around packet tracing and building packets by hand using low level programming. This also allows packet tracing to be introduced as support for debugging the student's own programs, following the Design-Implement paradigm of CDIO. Much of this detail is abstracted away by modern high level programming languages, and this can be problematic as it prevents students from developing the necessary knowledge to successfully troubleshoot

[2] `http://nsnam.org`

[3] In this context the 1999 Ericsson video, "Warriors of the Net" deserves special mention Warriors of the Net, Ericsson Media Lab 1999

network issues when they encounter them. This can be quite problematic, as the behaviour of a naively programmed application on a local, high speed network, can be quite different to the same application running over a wide area network due to issues of packet delay and fragmentation that may not be encountered under more ideal "laboratory" conditions. It is also the reality of many of today's internet based distributed programs that they incorporate their own software based networking in order to function, which has to solve many of the same problems as the underlying network equipment. By asking students to construct a peer-to-peer network in software, they are necessarily exposed to the reasons for many of the design choices underlying the creation of the Internet and today's distributed systems.

PROJECT BASED LEARNING

Whilst server resources were extremely limited, an important development here and elsewhere in the last decade is that all students can now be relied on to have laptops. Whilst dealing with vagaries in the different operating systems networking does present challenges, the ability to do so has always been an important, although not necessarily explicitly taught part of computer networking. To achieve our goals, we replaced the existing exercises with three consecutive, project based programming exercises which developed in complexity over the term:

1. A simple client-server program running between the student's laptop and a campus server.
2. A port knocking exercise using hand crafted packets to solve a series of server based puzzles.
3. Construction of a class wide peer-to-peer messaging network.

The first project was individual, but students were allowed to work in pairs for the second and third projects.

1: Client-Server programming

This assignment provided students with simple client/server messaging software and asked them to make small extensions to it. Developing network programs can be extremely frustrating for beginners simply due to the number of places things can go wrong, which are not limited to the student's code. Students need to learn to deal with the the vagaries of their local networking environment, and since unlike dedicated facilities laptops are prone to occasionally losing WiFi connectivity, the necessity of testing end to end connectivity is an extremely useful learning outcome. The main learning outcome of this exercise was to familiarise students with network troubleshooting, and the Berkeley sockets programming interface.

Network troubleshooting is perhaps the most valuable skill that we can attempt to impart in the general undergraduate networking course. Troubleshooting network issues can be a complex and demanding task. Even with simple undergraduate exercises, there are multiple points of potential failure besides the student's program. End to end connectivity in modern networks is challenged by features such as Network Address Translation(NAT), and security measures such as Layer 2 isolation which may invisibly block direct connections between student machines on the same WiFi segment. Whilst these can be frustrating to experience, they also useful provide reinforcement for key objectives in the course, and help to actualise what students are learning through lectures.

2: Port Knocking

Port knocking is a technique for restricting network access to applications by requiring a special sequence of packets to be sent to the application port, in order to establish full connectivity. In this exercise, which was developed by one of the courses TA's, Benedikt Þórðarson, students were asked to sent hand crafted UDP packets to a server, in order to unlock a series of puzzles, including the use of relatively obscure features of TCP/IP such as the "evil bit" (Bellovin, 2003).

The learning outcome of this exercise was to make lectures describing packet structures and protocols more relevant, and also to highlight key differences between TCP/IP and UDP, in particular with respect to packet loss. The puzzle element however, introduced a more interesting aspect to the exercise, whilst Instructor control over the target server made assisting with student debugging when necessary much easier, as server logs could be consulted to provide additional information.

Some technical challenges were encountered during this project, in particular it had to be designed around the behaviour of the campus VPN which blocked some packet settings. In later years we have been able to acquire a machine outside the campus firewall which avoids some of these issues, however non-standard firewall and VPN behaviour is increasingly becoming the norm, due to security considerations, so careful design and testing is recommended for these kinds of exercises.

3: Peer-to-Peer Networking

This was a class project, where students were asked to create nodes in a peer-to-peer TCP/IP based network, implementing a simple protocol to perform store and forward messaging between the nodes, allowing them to send short chat messages to other nodes on the network. In order to do this successfully, students needed to create a messaging layer using TCP/IP, which helped expose them to some of the underlying behavioural quirks of that protocol with respects to the arrival of bytes in the TCP/IP stream. TCP/IP guarantees ordered and reliable delivery of a stream of bytes, but makes no guarantees about the latency of their arrival. This can often create issues where networked programs work perfectly "in the lab" under ideal, uncongested conditions, but fail badly when deployed on more congested networks. This behaviour tended to occur naturally occur at the instructor nodes towards the end of the project, but modifications were also made to the instructor software to guarantee it in the last iteration of the project.

Students were provided with a simple protocol description, instructed that they could make extensions to it if they wished, and were encouraged to work together as a class to create as many network connections as possible between nodes, using the class piazza forum for discussions. Minor changes have been made to the protocol description every year to deter undue copy and pasting between the years.

Designing and implementing network protocols is not just a technical challenge, but on industry standards committees, can also be a non-trivial exercise in diplomacy and negotiation. Networking technology often offers choices between multiple solutions to a particular problem, and the realities that lie behind some design choices are not always apparent. In particular the challenge of getting changes accepted to existing protocols which have been widely deployed, are not always apparent to students. Part of the intention of the last project is to provide exposure to some of these issues surrounding protocol design. In the first, second and fourth years the project was set with a protocol that was deliberately incomplete, requiring students to reach a consensus on how to extend the protocol to resolve this. Since there were at least two different ways this could be done, this effectively created a class wide Fischer consensus problem (Fischer, Lynch, & Paterson, 1985) around agreeing which solution to pick.[4]

Classes have dealt with this issue in different ways. In the first year two competing standards were developed more or less simultaneously by two different groups within the class, who then actively and independently recruited other students to their standard. Attempts were also made to resolve the differences between the two approaches, but unfortunately this took the form of an extended debate about which was better. Although discussions on the class piazza forum were remarkably civilised, it is understood that the student's own facebook groups were slightly more heated. A number of student servers were developed that dealt with both protocols, and were awarded high marks for doing so. In the fourth year however, one student group spotted the issue with the description very quickly at the beginning of the assignment, wrote and circulated a detailed protocol extension with instructions on how to implement it, and also assisted other students with doing so. As a consequence the entire class adopted their standard.

Students are graded primarily on their implementation of the protocol, and overall code quality, but an ascending number of bonus marks are also provided for connectivity between groups, which provides an incentive both to provide continuously running servers, and to develop them as early as possible. This also helps to expose students to the critical importance in computer networking of extremely robust code that can run continuously for long periods. It also motivated students to write servers that were robust to minor variations in other student's implentations of the protocol, and to work with other student groups to resolve protocol issues - helping considerably in reducing instructor load for the project. Several instructor nodes were also provided running on a campus server, to act as backbone nodes for the network for connectivity purposes, and also to provide messaging load in the form of periodically broadcast MD5 hashes of messages that needed to be decoded. These nodes served as a known point that could be connected to by student nodes behind home network NAT protection, allowing these nodes to join the network, and then connect to other student nodes. The instructor nodes also provided useful output for debugging purposes, and some automated assessment.

DISCUSSION

It is easy when teaching computer networks to get lost in a dry discussion of the many data communication protocols used in the network stack, without managing to impart some of the broader aspects of protocol design and the technical challenges in actually implementing network communication mechanisms. Since many of these broader technical aspects are common to protocols at all levels of networking, asking students to create a software network provides practical experience that helps them implement some of the theories they are being exposed to. Developing techniques for routing messages between student nodes to maximise connec-

[4]This was not included in 2020, the first year of the Covid emergency, due to uncertainties around the success of this project in a purely online teaching environment.

tivity, determining methods to deal with nodes randomly entering and leaving the network, or sending malformed messages, helps reinforce and actualise the material they are receiving simultaneously in lectures. The traditional lecture material used for computer networks courses originated during a time when access to computer networks, even at university level, was very restricted. In Iceland at least, it is only in the last ten years that general availability of home network and laptop resources could be assumed in the student body. Now that these resources are available, it is possible to bring CDIO principles of active learning and design-implementation to the subject. Our experience is that this has dramatically improved educational outcomes and student satisfaction in the third year computer networks course. This was achieved without extra facilities simply by leveraging the student's own computers. Support software for the course was developed by the instructors.

Computer networking, one of the major contributors to the technological revolution of the last 50 years, should be an interesting and enjoyable course for students, even if its material can be complex. We believe the projects succeeded in part simply because they promoted student engagement with other students: building simple chat and messaging services in many ways parallels developments in the early days of the Internet, and students have described how they enjoyed chatting randomly with other students using the peer-to-peer network as they tested their software. This also helped to include students studying remotely, and effectively transformed the last weeks of a lecture based course into a virtual class collaborative effort.

We believe the general proliferation of cheap networked devices offers significant opportunity for further improving instruction in computer networking, and going beyond the examples in this paper, especially when combined with CDIO principles. We are adopting a similar approach for our computer security instruction, as are others(Buriachok, Sokolov, & Sokolov, 2020). Virtualisation also offers considerable opportunities to provide inexpensive but realistic networking workbenches (Yalcin, Altun, & Kose, 2015). Although peer-to-peer networking is generally considered an advanced topic in computer networking, constructing a simple class network with restricted capabilities has been consistently achieved by our students, and has provided a useful experience in practical networking for them.

ACKNOWLEDGEMENTS

We would like to acknowledge Benedikt Hólm Þórðarsson for his work in creating the second project, and Sigurbjartur Ingvar Helgason, Reykjavik University's Network Administrator, for his technical assistance and support when developing the projects described in this paper. We would also like to thank Cisco Networks for their generous support in building our campus network infrastructure.

FINANCIAL SUPPORT ACKNOWLEDGEMENTS

The author(s) received no financial support for this work.

REFERENCES

Bellovin, S. (2003, April 1). *The Security Flag in the IPv4 Header* (No. 3514). RFC 3514. RFC Editor. Retrieved from `https://rfc-editor.org/rfc/rfc3514.txt` doi: 10.17487/RFC3514

Buriachok, V., Sokolov, V., & Sokolov, V. (2020, 01). Implementation of active learning in the master's program on cybersecurity. In (p. 610-624). doi: 10.1007/978-3-030-16621-2_57

Dordal, P. (2020). *An introduction to computer networks*. Open Textbook Library.

Fischer, M. J., Lynch, N. A., & Paterson, M. S. (1985, April). Impossibility of distributed consensus with one faulty process. *Journal of the Association for Computing Machinery*, *32*(2).

Getchell, K., Miller, A., & Allison, C. (2005). A tcp learning environment. In *6th annual conference of the subject centre for information and computer sciences.*

Gupta, S., Ghonge, M., Thakare, P., & Jawandhiya, P. (2013, 04). Open-source network simulation tools an overview. *International Journal of Advanced Research in Computer Engineering & Technology*, *2*.

Gusev, M., Ristov, S., & Donevski, A. (2014, 04). Integrating practical cisco ccna courses in the computer networks' curriculum.. doi: 10.1109/EDUCON.2014.6826138

Kurose, J. F., & Ross, K. W. (2016). *Computer networking: A top-down approach* (7th ed.). Boston, MA: Pearson.

Malkin, G. S., & Harkin, A. (1998, May). *TFTP Option Extension* (No. 2347). RFC 2347. RFC Editor. Retrieved from `https://rfc-editor.org/rfc/rfc2347.txt` doi: 10.17487/RFC2347

Prvan, M., & OžEGOVIć, J. (2020, jun). Methods in teaching computer networks: A literature review. *ACM Trans. Comput. Educ.*, *20*(3). Retrieved from `https://doi.org/10.1145/3394963` doi: 10.1145/3394963

Sarkar, N. (2006). Teaching computer networking fundamentals using practical laboratory exercises. *IEEE Transactions on Education*, *49*(2), 285-291.

Tanenbaum, A. S., & Wetherall, D. (2011). *Computer networks* (5th ed.). Boston: Prentice Hall.

Yalcin, N., Altun, Y., & Kose, U. (2015). Educational material development model for teaching computer network and system management. *Computer Applications in Engineering Education*, *23*(4), 621–629.

BIOGRAPHICAL INFORMATION

Jacky Mallett is an Assistant Professor in the School of Technology, Department of Computer Science at Reykjavik University in Reykjavik, Iceland. She teaches Computer Networks and Computer Security using a project and puzzle based approach for class assignments. Her research focuses on complex networked systems, computer security, and simulation of the banking system.

Marcel Kyas is an Assistant Professor in the School of Technology, Department of Computer Science at Reykjavik University in Reykjavik, Iceland. He graduated from Christian Albrechts Universität zu Kiel in 2001, and received his Ph.D. from Leiden University in 2006. Previously, he taught at the University of Kiel, University of Oslo, and Freie Universität Berlin. His current research focuses on ambient assisted living, indoor positioning, and the design of safe and secure embedded systems. Lately, he got interested in sustainable computing, looking at the resource costs of software. He teaches in the form of project-based courses.

Stephan Schiffel is an Assistant Professor in the School of Technology, Department of Computer Science at Reykjavik University in Reykjavik, Iceland. He graduated from TU-Dresden with a Ph.D. in general computer game playing. His research and teaching focus is machine learning and general computer game playing.

Corresponding author

Jacky Mallett Reykjavik University
Dept. of Computer Science
Menntavegur 1, 102 Reykjavik
Iceland jacky@ru.is