EasyChair Preprint

№ 7613

A Bare Machine Tool to Learn System Internals in Computer Science Education

Joel Weymouth, Ramesh Karne, Alexander Wijesinha and Dheeraj Naraharisetti

March 28, 2022

A Bare Machine Tool to Learn System Internals in Computer Science Education

Joel Weymouth
Computer and Information Sciences
Towson University
Towson, MD USA
jweymouth@towson.edu

Ramesh K Karne
Computer and Information Sciences
Towson University
Towson, MD USA
rkarne@towson.edu

Alexander L Wijesinha
Computer and Information Sciences
Towson University
Towson, MD USA
awijesinha@towson.edu

Dheeraj N Naraharisetti
Marriotts Ridge High School, Marriotsville, MD
dheeraj.naraharisetti
@gmail.com

Abstract— ***Research in Bare Machine Computing has experienced significant changes over the years. As a result of the recent events of the pandemic and the need for remote learning, the Bare Machine Internet (BMI) as part of the Bare Machine Computing (BMC) paradigm experienced further studies. This research created tools to examine the internals of the protocols, memory, and scripts as a BMI became feasible. With the increased visualization in teaching Computer Science and remote learning, these features created with the BMI were considered efficacious in Computer Science Education (CSE) BMC/BMI is used in programming applications to run directly on a device. The paradigm is software that runs directly against the hardware using CPU, Memory, and I/O. The software application runs without an Operating System and resident mass storage. An essential part of the BMC paradigm is the Bare Machine Internet. It utilizes an Application Development model software that interfaces directly with the network and file servers' hardware. Because it is "bare," it is a powerful teaching and research tool that can readily display the internals of the network protocols, software, and hardware of the applications running on the Bare Server. The research demonstrated that the bare server was accessible by laptop and smartphone/android. The purpose was to show the further practicality of Bare Internet in Computer Engineering and Computer Science Education and Research. It also showed that an undergraduate student could use a bare server with any device and browser at any release version connected to the internet. This paper presents the Bare Web Server as an educational tool. We will discuss possible applications of this paradigm.***

Keywords— Bare Machine Computing, Online Research, Operating Systems Introduction

I. INTRODUCTION

This research and paper are to discuss the process of developing a bare application that runs on a bare web server for an educational tool. Innovation within computer science education is not new at all. Papers discussing innovation in education are not new and can be traced into the 1990s. For example, Oakley advocated the virtual classroom [13] and created the environment supporting the asynchronous classroom.[14]. Computer Science Education Research has evolved from traditional lecture pedagogy with lab-based learning to simulation game-based learning[11]. Further, the first paper that suggested the BMC paradigm by Karne was in 1995 [1]. BMC has been successfully demonstrated in several contexts, from database applications to web servers. The overall work concerning Bare Machine Computing will be related to this research later. In the meantime, Figure 1 shows a high-level presentation of the paradigm compared to conventional computing. Applying the BMC paradigm in further research and developing a learning tool was prodigious since lockdowns required a lot of remote research over the past year. Nirmala [18] suggested further study to leverage personal devices. With challenges, the research also uncovered possibilities and later avenues of research that would make the BMC paradigm a necessary and practical part of Computer Science Education. See Fig1, 2.

The paper highlights the challenges and findings of creating a learning tool on the bare web server and is organized as follows: Section 2, Bare Machine Computing and related work. Section 3, Educational Bare Tool Design and Implementation. Section 4, Demonstration of the educational tool, what this tool does. Section 5, Potential of the instrument in C.S. Education. Section 6, the significance of this research and further research; the broader impact of using bare devices in the classroom; section 7 Significance of this research and other research; the more general result of this work to use bare devices in the school. Section 8, Conclusions

II. BARE MACHINE COMPUTING AND RELATED WORK

The bare Machine Computing (BMC) paradigm is based on two fundamental principles. First, a computing device must be bare, and an application is written in the BMC programming methodology. It is an application and event-driven paradigm. A single or a set of end-user applications are bundled as a single monolithic executable, and applications directly communicate with hardware with no middleware. Fig. 1 illustrates such a concept. Traditional system calls provided by an operating system are replaced by application calls or direct hardware interfaces. A computing box is made bare, and there is no mass

storage onboard, no operating system running in the box. A given application or suite is a self-controlled, self-managed, and self-executed entity.

Application programs are made independent of execution environments, thus making them last for an extended period. End-user applications are derived using object-oriented abstraction to explode application space. Examples of applications include text processing, spreadsheets, email, server, browser, online meetings, payroll, etc. These applications are written in a single programming language development such as C/C++/Assembly, which can live for an extended period. As the operating system is eliminated, it tumbles all layers in current information technology and reduces obsolescence and inherent security. The computing device has no valuable resources to be exploited by an intruder. A given application suite is carried on a user-centric flash drive, has ownership, and can be run on any bare computing device. At this point, an Intel-based computing device is assumed to run bare applications. However, one can devise applications to run on any instruction set architecture.

The Bare Machine paradigm has some very distinct characteristics. A BMC user carries the required application suite on a removable device, such as a flash drive, and can connect to any available bare device without any ownership. Before the pandemic, the research and development were conducted exclusively in the BMC laboratory environment, locally hosting bare machines containing various application development projects. This BMC research has successfully produced many applications [2]-[11] on bare devices in the BMC laboratory, some of them being the following:

a) Bare Text Browser: BMC research produced a Bare Text Browser, a self-contained, self-managed, and self-controlled application handling memory management and a new API that communicates directly with the hardware and manages its threads.

b) VOIP (with IPSec): BMC research proved that Voice Over Internet Protocol (VOIP) did not need an operating system

c) Webservers (TLS): BMC Research has produced bare Web Servers that can communicate with each other. Researchers have developed a standard BMC Web Server module to work on a Bare Machine.

d) Split Servers (Protocol Split): BMC research did TCP protocol splitting using two bare servers. One of the servers was a connection server and a data server splitting the TCP protocol into its connection and data phases, executing these phases on different machines during a single HTTP request.

e) Email Client: BMC research developed an email server working on bare machines. This email server created was multi-threaded and allowed multiple users at any given time.

f) SQLite (Database): BMC research transformed and added SQLite to the list of applications that operate on a bare machine. The D.B. engine lends several levels of abstraction beyond the standard O.S. However, it is heavily dependent on system calls.

g) Multicore Web Server Architecture: The first BMC webserver was single-core and only 32 bit. Later the BMC paradigm migrated to multicore and is now compatible with 64-bit architecture.

h) IPSec: The IPSec baseline research suggested designing future versions of IPSec for high-performance or high-security bare P.C. applications and devices such as security gateways.

i) Gateways and Routers: BMC research has adapted APIs working with gateways and routers compatible on a bare P.C. using a six to 4 gateway, with IPv4 and IPv6.

These accomplishments make it possible to present the inner workings of operating systems, networks, protocols, binary executables, and devices in a manner that a student can easily understand. For example, from the memory dump in the application, students can learn how data is stored in memory and trace the actual value present in any memory address during debugging. BMC research can be a helpful teaching tool for computer science and engineering education. It dissects the executable and interacts with the bare device, which can be observed in real-time. As BMC applications avoid all middleware, there is a great potential for using this computing paradigm for educational purposes in computer science and related areas.

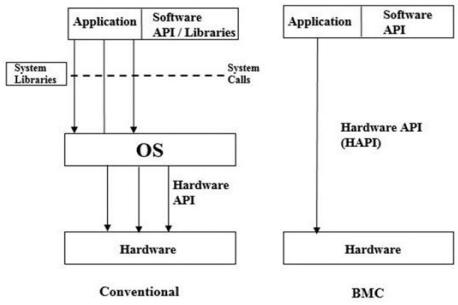

Fig. 1 Conventional and Bare Machine Computing

III. Educational Bare Tool (EBT): Design & Use

Educational Bare Tool (EBT) is motivated by the inherent simplicity in the BMC paradigm. Using this tool may be the easiest way to teach students the internals of a computer system. A student can understand the system internals (hardware and software) much easier, as the programmer has complete control of programming and execution of a given application suite. An existing Web server is modified to create the EBT

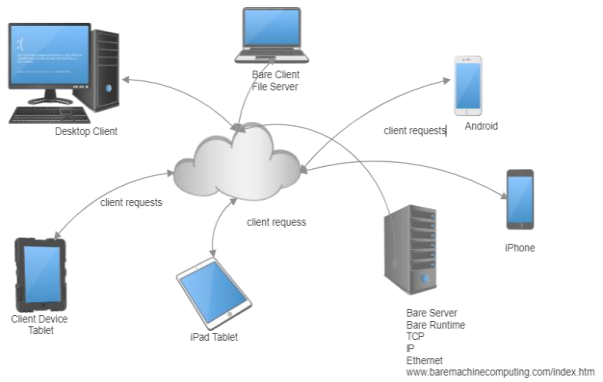

Fig 2 Bare Internet – the Server is a BARE Server

System. (Fig. 2) illustrates a system view of this tool. A BMC client, an O.S.-based desktop client, or an iPhone-based client can communicate with the bare Web server. Conventional client/server protocols communicate between a client and a server. Using the traditional HTML and PHP pages, EBT provides system internals (hardware and software) knowledge to a given student. This design mainly focuses on network protocol internals, task switching states (TSS), and memory dumps. A student can view a packet coming in from a network to internal details of the packet. All packets coming in are captured and stored in a circular list (receive list). All packets to be transmitted are stored in a circular list (transmit list). A student can view packets in the arrival order or transmitting order. A circular list is full wraps around the list for easy implementation. When a client request comes to a server, it will create a task and run this task until the request is complete. The request state and the CPU facilities are stored in TSS (task segment state). A student can view all general-purpose registers, stacks, program counter, and other parameters dynamically at the client site.

All client requests came as PHP or HTML page requests, satisfying ETHERNET, I.P., and TCP protocols. TSS, Memory, CODE, and the Receive/Transmit Descriptors. With a few modifications to the existing Webserver, the EBT was designed and implemented to provide learning capabilities for students in understanding network protocols and other system facilities. The enhanced code was written in C++ as the client requested special functions.

There is a great potential to enhance EBT to view more system internals. Notice, the code changes made in the Web server are minimal (~1200 lines of code), and the majority of the work involved was designing HTML and PHP pages. The simplicity and extensibility of bare Webserver make the design changes flexible and easier to test.

The Bare Server operates through a simple USB attached to a bare device powered up, with the bare file server to load all relevant files used in the tool. Considering the tool is a Bare Application, this approach applies to any Bare Application running on a bare server accessible to a client.

IV. The Tool In Practices

The tools start by opening a browser opening the site Baremachinecomputing.com/index.htm. (Fig. 3) A closer look at the page shows the six options. (Fig. 4). These six options: Ethernet Protocols (Fig 5), Internet Protocols (Fig 6), and TCP Protocols (Fig 7). Ethernet Protocols further examines the descriptors for receiving and transmitting the data packet. The TSS Code (Fig 12) and Memory Dump (Fig 13) discuss the internals of the memory-related descriptors.

A. Figures:

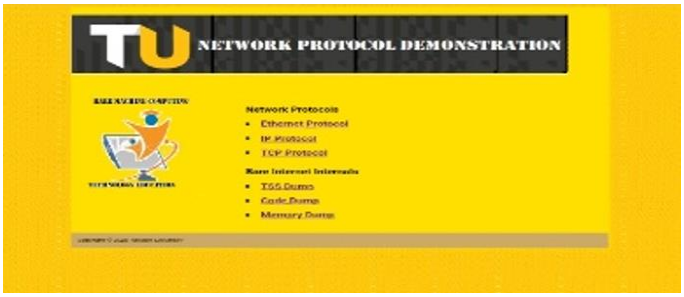

Fig. 3 EBT Main Menu

Fig. 4 Selection

Fig 5 Ethernet Protocol Header

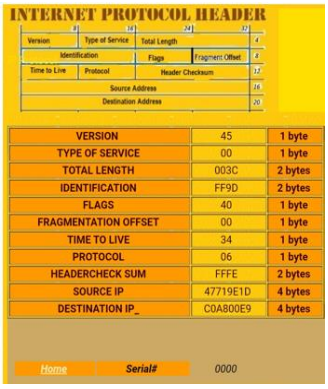

Fig 6. Internet Protocol Header

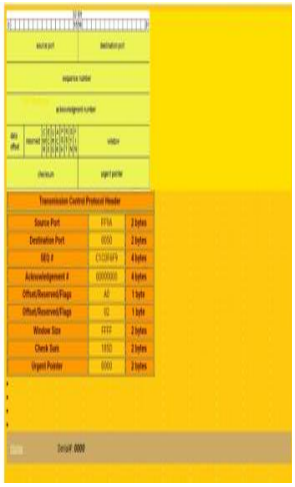

Fig. 7 Transmit Protocol

Fig 8 Trouble Shoot Script Internal

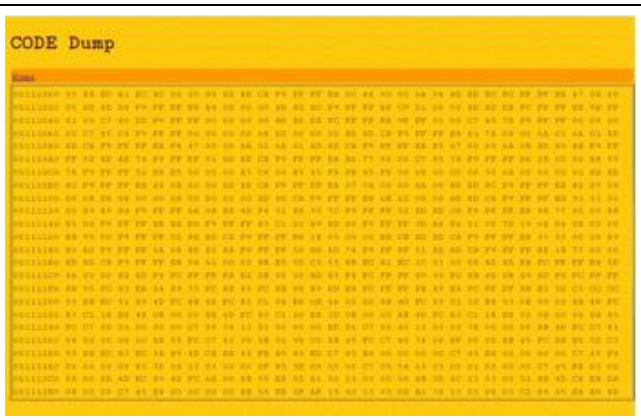

Fig 9. Bare Server Software Internals (Code Dump)

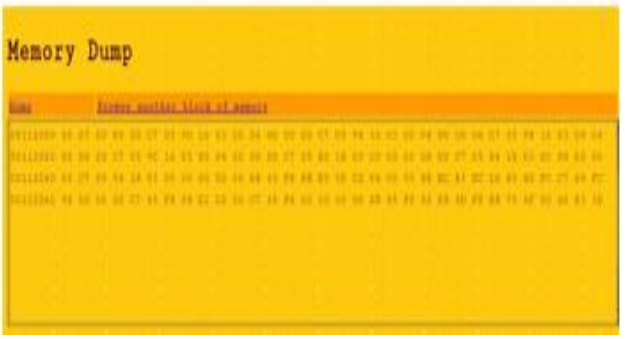

Fig 10. Internal Memory (Memory Dump)

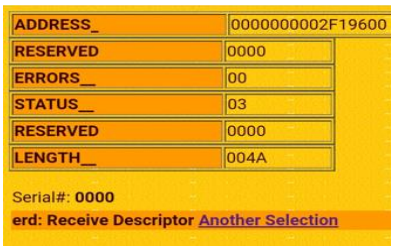

Fig 11. Receive Descriptor

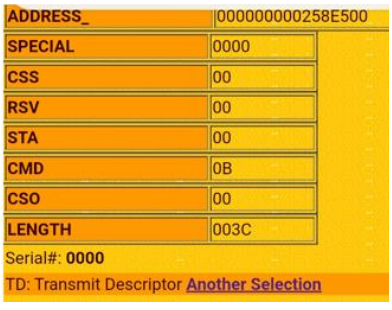

Fig 12. Transmt Descriptor

V. Potential of the tool in C.S. Education

Tools examining the internals of packets are not new. Wireshark accomplishes this [16] or MENeT [15] are just two examples of products that capture, filter, and display various packets from UDP, TCP, I.P., and Ethernet Protocol. The goal was not to build another packet sniffer. Instead, this research wishes to demonstrate BMC in Computer Science Education. Another major thing that sets this apart from Wireshark, is Ndatinya et al. demonstrated Wireshark as a Network Forensics tool, and the EBT is for education. [17]

The conventional impression of this tool is that it is a different flavor from Wireshark. Wireshark does: This tool does things that other devices like Wireshark do not perform. [16]

First, this tool identifies the Receive Descriptor List (RDL) and the network driver's Transmit Descriptor List (TDL). Many students do not know the existence of the RDL/TDL, but this tool introduces the concept to the Computer Science Student. (Fig 11, 12) The TDL is the data segment that is the link enabling the adapter to track transmitted packets in memory. Each sent packet requires one or more Transmit Descriptors. Receive Descriptors complete the same function but for the packets received. This feature takes the tool deeper than the data packet, exposing and dissecting the memory behind the network protocols. The device also looks at all the internal memory in detail with the Trouble Shooting Script Internals (see Fig 5), referred to as TSS Dump. Also a snapshot of the Bare Server Memory is available (see Fig 10) where the student may specify a specific address. The RDL and TDL screen has the address of each descriptor, so the student may examine where in memory these objects exist. Understanding the concepts of addresses in Computer Science is critical to problem solving and understanding system internals. There is also a way to see the bare internet software that interfaces directly with the hardware is also available for analysis. (see Fig 9).

The tool was introduced to several classes using Towson and Morgan State University labs. What was significant concerning these classes was that none of the students used the lab desktops. Table 1 shows how the students were using their devices. Table 2 shows the breakdown of student demographics. Interestingly, all of the IT Majors had heard of TCP, I.P., and Ethernet. None had ever heard of the TSS, TDL, or RDL. All were impressed that they did not have to install the tool. All I.T. majors had not considered their phones had a Mac Address and an I.P. address. The survey results are that all of the I.T. students had a greater interest in the internals of the Internet Operating Systems and a number of the non I.T. students interested in I.T. with two students changing their majors.

The conclusion was that system internal visualization and dissection of the client devices gave a greater desire further to investigate the architecture and intricacies of Computer Science.

TABLE I. Utilizaton of Devices

Device	***%***
Student Windows Laptop	70
Student MacBook	19
Student iPhone	9
Student Android	2
Lab Desktop	0

TABLE II. Student Demographics

Device	***%***
Freshman/Sophomore	75
Jr/Sr	25
IT Major	56
Business	25
Other	19

VI. Significance of this research and further research

A. The broader impact of this work is to use bare devices in the classroom

During this research, students made the following observations when using this tool.

The pressures of class and homelife creating a fast-paced and stressful environment makes moving to virtual or visualization learning paradigms helps them learn. Additionally, the effects of the past year with the pandemic and the unclear future of how assignments and classwork will be presented are helped by a more remote and virtual learning environment. Using their own devices without any special installation or configuration is a great plus because they are used to their own device, and there is no learning curve in using their own device. Finally, since bare internet works with even obsolete devices as well as Windows 10, or the latest Android or iPhone can support this tool, a student is not faced with an unexpected expense in order to use these tools. This gives the Computer Science Student greater flexibility to learn.

B. Modified a bare Web server to act as an educational tool

Several aspects of this toolset it apart from other tools. Beyond the apparent dissection of network protocols like Ethernet, I.P., and TCP, it looks at the bare server and the standard internet transactions. The student using their device can easily access this tool on the Bare Server. Therefore conventional clients can access the tool on a bare server. It allows the beginning student

introduction to the complexities "under the covers" of any operating system. It also introduces the early concept that operating systems are not confined to laptops, phones, and desktop computers but also to routers, bridges, hubs, and all aspects of the digital world in which they are about to embark. The more advanced student will see the internals of the memory behind the protocols and the TroubleShooting Script (TSS).

Modern undergraduates no longer rely on the computer lab within their university. They bring their device. The Educational Bare Tool does not require installation. Using the tool shows the diversity of each device. The EBT allows a student to observe and analyze the memory internals behind each network packet sent and received of the device. Users can see internal details of packets coming into the driver, interior components of software and hardware, and see the details can be seen and understood by the undergraduate. The basis of this tool can give a better insight into the internet, and it can dissect the internals much faster. Even though the server runs on a bare machine, it is accessible by conventional operating systems through an intranet and the internet.

VII. Conclusions

The utilization of this tool in undergraduate CSE should focus on improving its capabilities. Of course, visualization software development will be part of CSE for the foreseeable future. So will hardware using Operating Systems and Bare Machine Computing. Computer Networks and Network Security also will be part of the CSE curriculum. Therefore this will be a continuation of a new type of research, the dissection, and visualization of network protocols and operating systems as new and better approaches to presenting these subjects.

If this research proves promising, additional studies will examine the efficacy of a visualization/curriculum integration to the learner and study what makes the operating systems and networks, protocols for each level, testing, and possibly understanding learning milestones or grade levels. Ideally, the Software Engineering Architecture will enable the educator to customize games for each student. That part is leveraging long-established Information Technology concepts. Educators and software engineers must settle the variable factors like and implementation of the new paradigm.

References

[1] Karne, R.K., Object-oriented Computer Architectures for New Generation of Applications, Computer Architecture News, December 1995, Vol. 23, No. 5, pp. 8-19.

[2] He, L., Karne, R.K., Wijesinha, A.L., and Emdadi, A. A Study of Bare P.C. Web Server Performance for Workloads with Dynamic and Static Content, The 11th IEEE International Conference on High-Performance Computing and Communications (HPCC-09), Seoul, Korea, June 2009, p494-499.

[3] P. Appiah-kubi, R. K. Karne, and A. L. Wijesinha. A Bare PC TLS Webmail Server, International Conference on Computing, Networking nd Communications, ICNC 2012, Maui, Hawaii, January 2012, p.156-160.

[4] P. Appiah-kubi, R. K. Karne, and A. L. Wijesinha. A Performance Study of Conventional and Bare P.C. Webmail Servers, The Seventh International Conference on Networking and Services, ICNS 2011, p.280-285.

[5] P. Appiah-kubi, R. K. Karne, and A. L. Wijesinha. The Design and Performance of a Bare P.C. Webmail Server, The 12th IEEE International Conference on High-Performance Computing and Communications, AHPCC 2010, Sept 1-3, 2010, Melbourne, Australia, p521-526.

[6] N. Kazemi, A. L. Wijesinha, and R. Karne. Design and Implementation of IPsec on a Bare P.C., 2nd International Conference on Computer Science and its Applications (CSA), 2009.

[7] A. Alexander, A. L. Wijesinha, and R. Karne. An Evaluation of Secure Real-Time Protocol (SRTP) Performance for VoIP, 3rd International Conference on Network and System Security (NSS), 2009.

[8] G. H. Khaksari, A. L. Wijesinha, and R. Karne. Secure VoIP using a Bare P.C. 3rd International Conference on New Technologies, Mobility, and Security (NTMS), 2009.

[9] Ford, G.H., Karne, R.K., Wijesinha, A.L., and Appiah-Kubi, P. The Performance of a Bare Machine Email Server, 21st International Symposium on Computer Architecture and High-Performance Computing (SBAC-PAD 2009), IEEE / ACM Publications, 28-31 October 2009, Sao Paulo, SP, Brazil, pp. 143-150.

[10] Ford, G.H., Karne, R.K., Wijesinha, A.L., and Appiah-Kubi, P. The Design and Implementation of a Bare P.C. Email Server,33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC 2009), Seattle, Washington, July 2009, p480-485.

[11] B. S. Rawal, R. K. Karne, A. L. Wijesinha. Split Protocol Client Server Architecture, Seventeenth IEEE Symposium on Computers and Communications (ISCC'12), July 1 -4, 2012, Cappadocia, Turkey.

[12] B. Jong, C. Lai, Y. Hsia, T. Lin, and C. Lu, "Using Game-Based Cooperative Learning to Improve Learning Motivation: A Study of Online Game Use in an Operating Systems Course," in IEEE Transactions on Education, vol. 56, no. 2, pp. 183-190, May 2013, doi: 10.1109/TE.2012.2207959.

[13] B. Oakley, "The virtual classroom: at the cutting edge of higher education," Technology-Based Re-Engineering Engineering Education Proceedings of Frontiers in Education FIE'96 26th Annual Conference, Salt Lake City, UT, USA, 1996, pp. 135-139 vol.1, doi: 10.1109/FIE.1996.569928.

[14] B. Oakley, "Helping faculty develop new asynchronous learning environments," Technology-Based Re-Engineering Engineering Education Proceedings of Frontiers in Education FIE'96 26th Annual Conference, Salt Lake City, UT, USA, 1996, pp. 659-662 vol.2, doi: 10.1109/FIE.1996.573039.

[15] N. A. Junejo, N. Ahmed and A. Q. K. Rajput, "Network layer packet analysis using MENeT," *7th International Multi-Topic Conference, 2003. INMIC 2003.*, 2003, pp. 151-156, doi: 10.1109/INMIC.2003.1416681

[16] Shaoqiang Wang, Dongsheng Xu and ShiLiang Yan, "Analysis and application of Wireshark in TCP/IP protocol teaching," *2010 International Conference on E-Health Networking Digital Ecosystems and Technologies (EDT)*, 2010, pp. 269-272, doi: 10.1109/EDT.2010.5496372.

[17] Ndatinya, Vivens & Xiao, Zhifeng & Manepalli, Vasudeva & Meng, Ke & Xiao, Yang. (2015). Network forensics analysis using Wireshark. International Journal of Security and Networks. 10. 91. 10.1504/IJSN.2015.070421.

[18] N. Soundararajan, J. Weymouth, R. Karne, A. Wijesinha and N. Ordouie, "Remote Collaboration Potential in STEM Education using Bare Machine Computing Research," in 2020 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2020 pp. 872-878.doi: 10.1109/CSCI51800.2020.00164 Url: https://doi.ieeecomputersociety.org/10.1109/CSCI51800.2020.00164