Networking Education: A Caribbean Perspective

Daniel T. Fokum
University of the West Indies
Kingston, Jamaica
daniel.fokum@uwimona.edu.jm

ABSTRACT
This paper presents three networking courses that are designed to be taken in sequence by second-year (junior) and final year students in a baccalaureate Computer Science programme in the anglophone Caribbean. The first course is core for all Computing students, while the last two are electives. In this paper we present the rationale for this arrangement, the content for these courses, and make some arguments on how networking should be taught in the post-COVID era.

CCS CONCEPTS
- Social and professional topics → Computing education;
- Networks;

KEYWORDS
Networking Education, Managing Enrolment Growth

ACM Reference Format:

1 INTRODUCTION
The purpose of this paper is to present the author’s experiences with teaching computer networking courses in a blended mode in the pre-Covid pandemic era, and to use these observations to shape some recommendations on how computer networking ought to be taught in the post-COVID era. Beginning in the 2012/13 academic year The University of the West Indies, Mona– the Jamaican campus of a regional research university–introduced a new set of networking courses. Prior to this point, there was only one networking course in the entire curriculum. This course was a final year elective that was taken by about half of all Computer Science students. In response to the ACM/IEEE-Computer Society Computer Science Curriculum 2008 guidelines [2] my department introduced a core course called Net-Centric Computing. This course contained all the core and some of the elective units of the Net Centric content area in those guidelines. Since 2015 this course has been taught in a “blended” mode, where most of the students are present in Kingston, with other students at a remote campus in Western Jamaica. In 2017 and 2018 the course was also taught to students in Barbados. In Section 2.1 we present the course content in greater depth.

Given that there are many other networking concepts that are not covered in Net-Centric Computing, a second course called Principles of Computer Networks was introduced. This course is a final year elective for all Computing students. In this course, students learned about other networking topics such as congestion control, routing, and emerging areas such as Software-Defined Networks. In Section 2.2 we present the course content in greater detail.

In Jamaica many of the ads for networking positions frequently ask for bachelor’s degrees in Computer Science and some certification like the CCNA. As a result, the third course, Implementation of Computer Networks, was created. The goal of this course was for students to run and tune different networking protocols on emulated hardware. Recently the department received a donation of networking gear that was being replaced by a Jamaican company. The plan was to use this hardware at the next offering of the course, which will probably be in about two years time, subject to the number of students selecting the course. In Section 2.3 I present an overview of the course.

1.1 Assumptions about Prior Knowledge
Undergraduate degrees in the anglophone Caribbean are based on the UK model. This means that our degrees take three years, but students arrive at university with narrower and slightly more advanced qualifications than their American counterparts. Many Computer Science programs in the UK traditionally require good grades in Advanced Level Mathematics. However, in the 2010/2011 period my department opted to require passes in any CAPE (Caribbean Advanced Proficiency Examination) [Similar to the Advanced Level exam in the UK] Science subject (Biology, Chemistry, Computer Science, Physics, or Pure Mathematics) with the expectation that our programme would provide any mathematical content that was necessary.

The first two courses in our sequence use “Computer Networking: A Top-Down Approach” [4] as the primary textbook. Given that the course does not use much advanced Mathematics beyond what is taught in high school, our treatment of networking does not assume knowledge of statistical methods or Calculus.

The rest of this paper is laid out as follows: Section 2 presents the structure of our networking courses. In Section 3, I present some observations from teaching the networking courses and discuss how these experiences might apply to teaching computer networking in the post-COVID era. Section 4 concludes the paper. One of the observations I make is that the broader Computer Science education community already shares CS1 projects, and therefore, the computer networking education community should also move in...
that direction. In the appendices I share two programming projects that I have developed as my contribution to project sharing.

2 COURSE STRUCTURES

In this section I provide details on the structure of all the networking courses.

2.1 Net-Centric Computing (COMP2190)

The Net-Centric Computing course has been taught nine times; approximately once per year since 2013. Over this period the enrollment in the course has gone up from about 70 to 200 students, which accounts for over 150% growth. Beginning in 2014 the course has also been taught in blended mode to students who were in Western Jamaica or in Barbados. The course is delivered using three hour-long lectures and an hour-long tutorial. The tutorials are like discussion sessions in the US. Each tutorial requires students to work through several problems with the help of a “tutor.”

Net-Centric Computing has as pre-requisites four courses: our Introduction to Computing course sequence—that is taught using Python,—Object-Oriented Programming—taught using Java,— and a Mathematics course developed for Computing majors. In the latter course, students learn about probability measure, events, conditional probability, and independence amongst other things. The pre-requisite courses allow us to teach socket programming using Python without having to teach any additional Python syntax beyond what is needed for socket programming. In addition, the background from the Mathematics for Computing course allows for simplified mathematical treatment of the benefits of packet over circuit switching.

As mentioned in Section 1, COMP2190 uses [4] as the textbook. The content is as follows:

**Introduction** Introduction to Computer Networks and the Internet. We cover all the sections of [4, Chp. 1] to set a foundation for the course

**Application Layer** Students are introduced to the application layer, HTTP and its message format, DNS, and then get an introduction to socket programming.

**Transport Layer** Students are introduced to the application layer, HTTP and its message format, DNS, and then get an introduction to transport layer, demultiplexing and multiplexing, UDP, TCP’s segment structure, round-trip time estimation, reliable data transfer, flow control, and connection management. Observe that there is no discussion of congestion control. This is deferred to an advanced course.

**Network Layer** Students learn about the network layer and its services, routing versus forwarding, the Internet protocol and the IPv4 datagram format, and IPv4 addressing. Students also get an introduction to IPv6. Note that there is no discussion of routing in this course beyond a definition of what it is.

**Security in Computer Networks** Students learn about what network security is, symmetric key cryptography, public key cryptography with an emphasis on the RSA algorithm, message integrity and digital signatures, cryptographic hash functions, message authentication codes, digital signatures, IPSec and VPNs, firewalls, and intrusion detection systems.

**Web programming** Students learn about XHTML, CSS basics, JavaScript basics and some simple server-side scripts. This portion of the course is typically done in three weeks, so the treatment of the web topics is quite superficial.

2.1.1 Assessment. The COMP2190 course is assessed with a combination of quizzes, programming projects, written homework assignments, and exams. From the course’s inception quizzes were administered whenever we finished covering one layer of the protocol stack. These quizzes were initially administered through the learning management system (LMS). However, beginning in 2015 those quizzes were administered weekly in class, with students receiving at least two different versions of the quizzes to discourage cheating. Growing course enrollments led to the quizzes being administered using Plickers last year. However, given the class size it took several sweeps to scan the entire class. It is anticipated that with our university transitioning to online teaching for at least the first semester of the upcoming academic year, these quizzes will again be administered via the LMS. This time students will receive randomized, but otherwise identical questions to discourage cheating.

The programming projects are specified in Appendix A. The first project always involves a socket programming exercise, where students have to develop a simplified client and server to transfer some information. Beginning in 2015 this project has always implemented a simplified text-based protocol to reinforce the content from class on protocols. The second project always involves students implementing simplified RSA to secure key exchange between a client and server. As before, this is intended to reinforce the lecture content for students. Appendix B shows the specification of one of the cryptography assignments. The last programming assignment is always a web programming exercise, that involves three smaller exercises: 1) formatting some static HTML with CSS; 2) building an HTML form and performing client-side validation with JavaScript; and 3) connecting the HTML form to a database, and doing some server-side validation prior to persisting the data. All three projects have always been marked manually, but in the upcoming year the plan is to use test scripts to autograde student submissions, certainly for the first two projects.

The first homework assignment always involves students using common network command-line tools such as ping, traceroute, arp, netstat, etc. to perform tasks like testing reachability of a given interface or the path that a packet follows between two interfaces. The second assignment involves students using given Wireshark trace files to answer a series of questions about different protocols.

2.2 Principles of Computer Networking (COMP3191)

The Principles of Computer Networks course has been taught seven times; once per year since 2013. Unlike the first course, which is core for all Computing students, this one is a final year elective, and this is reflected in the enrollment. The enrollment has ranged between
20 and 35 students, with the number of students fluctuating from year to year. The pre-requisite for the Principles of Computer Networking course is Net-Centric Computing. COMP3191 is structured using the same mix of lectures and tutorials as the Net-Centric Computing course. COMP3191 uses [4] as the textbook. The content always includes the following topics:

**Architectural Principles** A review of what the Internet is, encapsulation, and an introduction to ideas such as the fate-sharing and end-to-end principles, and the “hourglass philosophy.”

**Application Layer** Students learn about HTTP caching, HTTP/2, DNS name compression, SMTP, FTP, and BitTorrent.

**Transport Layer** Students learn about how TCP implements reliability using state machines, congestion control, and TCP’s fairness.

**Network Layer** Students get a brief review on the network layer and then we learn about distance-vector and link-state algorithms, followed by treatment of RIP, OSPF, and BGP, and finally multicast routing.

**Link and Physical Layers** Students learn about modulation and coding, Shannon’s capacity theorem, error detection and correction (including CRCs, checksums, and parity checks), why Ethernet imposes limits on packet and network size, spanning tree protocol, VLANs, what makes wireless different, 802.11’s rate adaptation, and adaptive modulation and coding in 802.11.

The following topics have also been covered in one year or the other:

**Software-Defined and Data Center Networking** Students get an introduction to software-defined networking (SDN), match-plus-action, and protocols to enable SDN. Students also learn about how data center networking works and a contemporary view of some of the issues with data center networking.

**Multimedia networking** How to stream stored audio and video, and coverage of DASH (Dynamic Adaptive Streaming over HTTP).

2.2.1 Assessment. The COMP3191 course is assessed with a combination of quizzes, programming projects, written homework assignments, and exams. The quizzes for COMP3191 have evolved just like the quizzes for COMP2190. Last year the quizzes were administered using Plickers, and students liked the quick feedback.

There have always been two programming projects. The first programming project has always involved simulation of a reliable data transfer protocol, while the second project involves writing code to simulate a distance vector protocol. In this course I have not relied on my own projects. Instead, I have used projects such as those from [6] or others with permission. One of the assignments in this course has been to for students to argue for or against a position on some contemporary network-related issue. For example, a few years ago some Jamaican telcos blocked traffic from some Voice over IP applications. Students are then supposed to argue for or against the carriers’ position.

2.3 Implementation of Computer Networks (COMP3192)

The Implementation of Computer Networks is another final year elective course that has been taught twice since 2015. Less than 12 students have signed up for the course since 2018, so it has not been offered. The point of the course was for students to get experience configuring different network protocols in a sandbox environment. By the end of the semester students are simulating routing within an AS and between ASes using OSPF/RIP and BGP respectively. The pre-requisite for the Implementation of Computer Networks is Principles of Computer Networking. This dependency exists because routing protocols are treated in the Principles of Computer Networks course. However, a handful of students have been allowed to advance into Implementation of Computer Networks without taking the Principles of Computer Networking course. Given that the department does not plan to offer COMP3192 in the upcoming year as part of a plan to reduce elective course offerings, I will not discuss the course content or its assessment further.

3 OBSERVATIONS AND DISCUSSION

Over the period that the courses have been taught, I have noticed that there is much greater student engagement with the Principles of Networking course versus the Net Centric Computing course. A common complaint amongst students is that the Net Centric Computing course moves fast. In fact, this was a secondary reason for moving to administer weekly quizzes, so that students study consistently throughout the semester. In the first year after introduction of the weekly quizzes there was a statistically significant improvement in student performance in projects and exams. Unfortunately, in recent years the weekly quizzes have not had the desired impact on student performance, with an increasing percentage of students achieving poorer grades. I began putting videos of lectures online in 2016, and this appeared to coincide with a decline in grades. A more thorough study needs to be carried out to determine the reasons for the drop in performance.

Beginning in 2017, when I had to teach students in both Jamaica and Barbados, I began to create videos of lectures to put on the LMS. The videos are short, i.e., typically less than 20 minutes, mini-lectures on different course topics that are posted to YouTube and linked in the LMS. The expectation was that students would watch these videos prior to coming to class, so that the class time could be used for solving problems or addressing students’ questions from the videos. In practice, I have observed that few students watched the videos before class, with the number of video “hits” going up significantly whenever students have projects or assignments due. In January 2019 we were able to get log files from the LMS for COMP2190 covering most of the semester. In the LMS most of the videos were shown to the students as YouTube links. Fig. 1 shows how students clicked on the links during the semester. The dashed horizontal line in Fig. 1 represents the mean number of URL accesses for Net Centric Computing in 2018. The very pronounced peak in Fig. 1 occurs around the time when students had their midterm. Unfortunately, the log file ended three days before the final exam, and so we do not have the matching peak for the final exam. However, review of the channel views showed an even more pronounced jump in the number of views.
Academic integrity is a major issue that instructors will have to contend with in the post-Covid reality, especially as a lot of teaching and assessment will probably occur virtually in the short term. This will mean that instructors will have to come up with relevant/original assignments that test networking education. Developing these types of assignments from one year to the next can be hard, and so it calls for exchanging pedagogical material. The broader CS Education community has the Nifty Assignments session at SIGCSE [5] as a way to trade assignment ideas and materials. It will be good for the networking education community to move in that direction. Even with original assignments, there is also the need to test student submissions with plagiarism checkers, e.g., MOSS [1]. This becomes even more important if students are working on projects that have been traded with other instructors.

Another academic integrity issue has to do with exams. To defeat cheating on exams it is desirable for students to have different versions of the same exam. Ideally, students should have exam questions that are randomly generated, but test the same set of concepts at similar levels of difficulty [3]. This is somewhat easier to achieve for questions involving calculations or operations on graphs, and harder on questions that involve synthesizing information, or identifying which properties might not apply to a protocol. In the short term, there should be some collaboration on creating a test bank that network educators can use for assessment.

The lack of Internet penetration and/or the cost of access are major issues that one will have to contend with as teaching moves online. According to Statista, Jamaica had an Internet penetration of 55% in 2017. In addition, some of the students who might ordinarily have Internet access while living in Kingston (the capital) do not have access at their homes in rural areas. This inequity will persist until the telcos can expand their reach in rural areas. With regards to the cost of Internet access, the Guild of Students (student government) at my university approached the Ministry of Education of access. Ultimately, only one carrier came on board. This carrier provided a list of domains that students could access at no charge, subject to fair-usage rules and students holding an “Educational plan” with the carrier. In reality, it is quite hard to implement this no charge access. For example, my institution uses an open-source LMS and a commercial platform for holding classes online. The LMS is hosted on the University’s domain, whereas the commercial platform is hosted in the cloud in an overseas datacenter. Given that the commercial platform is in the cloud, it is hard to identify it for the no-charge access, because that same cloud provider is used to host other commercial products. In addition, some courses had recordings of lectures that are hosted on YouTube. Each time a student attempts to access that recording, that retrieval will be metered, because the YouTube domain was not one of those with no-charge access. Some colleagues attempted to get around the no-charge access issues by hosting all of their lecture videos on the local LMS, however, this solution is sub-optimal because the local LMS is not optimized for streaming content. Furthermore, this approach can potentially lead to a bottleneck at the university’s access links.

Given some of the issues that we have observed with Internet access, my department will be delivering instruction mostly asynchronously in the upcoming semester. In the case of the networking courses, this will mean that lectures will be pre-recorded, ideally in short (less than 30 minutes) chunks. During the scheduled class time we will solve problems and answer questions. These sessions will also be recorded and made available to students so that those who might either not have access to devices or the Internet at the scheduled class time can still benefit from instruction.

It is worth noting that issues around Internet and device access are not limited to Jamaica or the Caribbean. Such issues are probably quite prevalent in other developing countries. In addition, issues of Internet access might also affect students in developed countries, particularly those living in rural areas.

Finally, while the networking courses have not yet been taught in the pandemic era, colleagues have shared the following observations from teaching in this era:

- Attendance dropped in lectures. It is possible that this is because students knew that lectures were recorded and they could watch them at their leisure.
- Some students explicitly had issues coping with classes online. This made it difficult for them to concentrate on the subject matter during class. It is likely that this was a function of the restrictions at that time of the pandemic. In the early days of the pandemic Jamaica was subject to a work from home order, and there was also a 6 PM to 6 AM curfew.
- The students in second year, i.e., those who would have taken the first networking course, appeared to be more comfortable asking questions. The more advanced students were quite comfortable asking questions via their microphones.
- The use of polls in the online environment were effective at checking whether a point made in the lecture was understood. Generally, students responded to the polls better than they did to open questions (whether responding by text or by voice).

These observations indicate that some effort needs to be expended to create more engaging lecture content. In addition, one needs to create poll questions to use in the middle of lectures. In addition, institutions, particularly those in developing countries, will need
to explore means to ensure that no students are left behind because they are unable to get online to access content.

4 CONCLUSION

In this paper we have presented the content for two networking courses and made some claims for how computer networking should be taught in the post-Covid era. One of the takeaways is the need for greater collaboration between educators. This collaboration can be in the form of shared programming assignments, as the broader CS Education community does with Nifty Assignments, and in the form of a shared question bank. In the appendices I have shared two programming assignments as a way to start the discussion on sharing projects. I have also shown that while instructors may favor a “flipped classroom” approach with pre-recorded lectures, many students may not watch the videos beforehand. Instructors will need to grapple with how to engage students better, so that they can watch the videos before class.

REFERENCES


Figure 2: Message sequence exchanged between client and server

A SOCKET PROGRAMMING

In this assignment, you’ll write a client that will use sockets to communicate with a server that you will also write. Your client and server will implement a very simple Diffie-Hellman key exchange. Your client and server will exchange the sequence of messages shown in Fig. 2.

Your server will create a string containing its name (e.g., “Server of John A. Smith”) and then begin accepting connections from clients. Your client should first accept two integers from the keyboard—one a prime number, \( p \), between 257 and 1021, and the other, \( g \), called a generator that is strictly less than the prime. Your program will ensure that these numbers satisfy the relationship that \( g^i \mod p \) can generate all integers between 1 and \( (p - 1) \) for all \( i \) between 1 and \( (p - 1) \). Some pairs of numbers \((g, p)\) for which this is true include \((3, 257)\), \((5, 743)\), \((11, 769)\), \((15, 907)\), etc. Once you have collected these numbers and they are valid, open a TCP socket to your server and send a message containing the string 110 Hello and then wait for a server reply. The server should reply to your message with the string 110 Hello. After the server acknowledges the client’s “Hello”, the client should send a packet with the string 110 Generator: concatenated with the generator integer, which is in turn concatenated with string Prime: followed by the prime. The server will acknowledge this message with the string 111 Generator and Prime Rcvd. The client and server will then
separately compute their private and public keys by calling the appropriate methods, which are provided for you. The client should then send a packet with the string 120 PubKey: concatenated with the client’s public key. The server will acknowledge this packet with the string 120 PubKey: followed by the server’s public key. Next, the client will generate a random integer, a nonce, and encrypt it with the server’s public key. The client will send a packet with the string 130 Cipher text: followed by the encrypted nonce. The server will extract the encrypted nonce, decrypt it, subtract 5 from the number and then encrypt the transformed nonce with the client’s public key. The server will send a packet with the string 130 Cipher text: followed by the transformed and encrypted nonce. Once the client receives this message, it should print out a status message of 200 OK if the absolute value of the difference between the decrypted nonce and the original nonce is 5. Otherwise you should print 400 ERR to the console. At this point, both the client and the server should print out q, p, their respective public keys, their respective private keys, and the other process’s public key. The client then terminates after releasing any created sockets.

The last part of the assignment should be a fun social thing to do. Team up with someone from the class and get your client to interact with their server, or vice versa. If you’ve got a server running, you can advertise your server’s services to anyone in the class, and if you’ve got a client, all you need to do is interact with such an advertised server.

Note that if you have got your own client and server running, there isn’t any more programming involved—just running your client and server with someone else’s. This shouldn’t actually be very hard at all. For this part of the assignment, you need to hand in the output from both the client and the server. On the Internet the IETF requires that two independent implementations of a protocol must interoperate; that’s what you are demonstrating here.

Note that a short design document is required. You should program each process to print an informative statement whenever it takes an action (e.g., sends or receives a message, detects termination of input, etc.), so that you can see that your processes are working correctly (or not!). This also allows the grader to also determine from this output if your processes are working correctly. You should hand in screen shots (or file content, if your process is writing to a file) of these informative messages.

B CRYPTOGRAPHY

In this assignment, you will gain better understanding of cryptography by implementing a simplified version of RSA encryption and then by using public keys generated by the software, PGP.

In the first part of the assignment a server process will send its public key to a client process. The client process will then use the public key to send a short session key to the server. The server will decrypt that key, which will eventually be used in a symmetric encryption implementation. The second part of the assignment involves you generating public and private key pairs to send messages to each other with confidentiality.

B.1 Simplified RSA Implementation

As we discussed in class, RSA encryption is computationally expensive and so it is typically used to distribute session keys for a symmetric encryption algorithm. In this assignment we will simulate the existence of two parties, a client and a server, that use public key encryption to exchange a session key that will be used to encrypt data using the simplified AES algorithm.

At program start-up, the client will send a message 100 Hello to the server. The server will respond to this message with 101 Hello n e, where n and e represent the modulus and exponent, respectively, of the server’s RSA public key. Upon receipt of this message, the client will send the message 110 SessionKey s, where s is a symmetric key for this session that has been encrypted with the server’s public key. The server will then respond to this message with 120 Nonce r, where r is a randomly chosen integer that is less than n and encrypted with the server’s private key. The client will respond to this message with 130 s(r), where s(r) represents the encryption of r using the symmetric key and the simplified AES algorithm. Upon receipt of this message the server will decrypt s(r). If the server is able to retrieve the correct pseudorandom string, the server will send the client the message 200 OK, otherwise it will send the message 400 ERR.

B.1.1 Server Implementation. Write a function that receives as its parameters primes p and q, calculates public and private RSA keys using these parameters, and outputs n, e (Euler’s totient), d, and e as printouts to standard output. Note, that you must use the Extended Euclidean Algorithm to compute d. Write another function that encrypts/decrypts data using RSA. That function will accept as inputs three integer parameters for the message, modulus, and exponent, respectively. This function should encrypt/decrypt the input message using the other parameters and return the result. This function must make use of the modular_exponentiation function, whose pseudocode is specified for you.

When you start up your server you will accept two prime integers whose product is between 56000 and 65536 and compute the public and private keys using the function defined above. After the client says “Hello” the server will respond to the client’s message as described above. The client will then send a 16-bit integer, a session key, that is less than n and encrypted with the server’s public key. The server will retrieve this message by applying its private key. Next the server will respond to the session key message by sending a 16-bit pseudorandom integer, a nonce that is encrypted with the server’s private key. The client will decrypt the message by applying the server’s public key. The client will then encrypt the nonce using the simplified AES algorithm and send the encrypted nonce to the server. The server will decrypt the message by using the simplified AES algorithm and the session key. Finally, the server will compare the decrypted message with the nonce that was sent. The server will respond as specified above, and then close its socket.

B.1.2 Client Implementation. You should write a function that takes three integer parameters for the plaintext, modulus, and exponent, respectively. This function should encrypt the plaintext input using the other parameters and return the result. This function must make use of the modular_exponentiation function, whose pseudocode is specified for you. You should also write a function that generates a random session key, that is at least 15 bits long and no more than 16 bits long. After the server responds to the client’s hello message, the client will send this session key encrypted with
the server’s public key. The server will respond with a pseudo-random 16-bit integer that is encrypted with the server’s private key. The client will decrypt this integer and retrieve the pseudorandom string. Your decryption function must also make use of the `modular_exponentiation` function. Next, the client will send this pseudorandom string encrypted with the session key that was sent to the server earlier. Finally, the client will listen for the server to send back a message as to whether everything is OK. Once it receives this final message, the client will close its socket.

For debugging purposes you should print out all the messages received from the server. Make sure to document clearly in your code any assumptions you make about the input and encryption algorithm.

**B.2 Extra credit**

The RSA algorithm depends on the difficulty of factoring large numbers. The 16-bit or 17-bit keys that we use in Part C.1 can be easily factorized on your computers. If you get a status code of `200 OK` in part C.1, write a function that takes as input the server’s public key and then uses that to compute the server’s private key. Print out the server’s private key to standard out. If you attempt this problem your write-up should also include a sketch of a solution to prevent an adversary from getting a victim’s private key given the public key.