Term Project (45% of Course Grade)
Assigned: Monday, March 13, 2017

CS 4516 - Advanced Computer Networks

Project: DNS Capabilities via Software-Defined Networking

Through a series of project phases, CS 4516 students will incrementally build a software-defined network (SDN). An SDN allows a network administrator to control packets arbitrarily, without being restricted by the firmware in network switches or routers. We will begin with preliminary configuration of the network. We will then create a basic system and then use the SDN functionality to re-create the DNS capabilities system proposed by Shue *et al.*[1].

The class will implement each aspect of the SDN, including the SDN controller, the control protocol, and the agent software on the controlled components. Unlike in traditional networks, in which SDN agents run on network infrastructure (such as routers and managed switches), the SDN controller in this project will be commanding agents running on Linux virtual machines (VMs). These agents will alter the host configuration on the controlled systems to enact new network configurations.

In this project, the class will develop a deeper understanding of computer networking across layers 2 through 5 while learning about the latest networking innovations, including software-defined networking. The class will have roughly one week to complete the first phase of the project and the remaining three phases will each last slightly under two weeks.

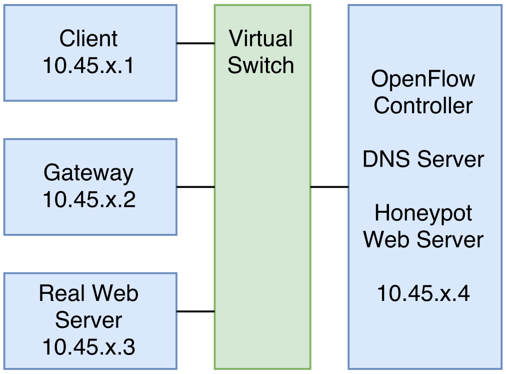


Figure 1: Basic network diagram for the CS 4516 term project.

The Big Picture

In this project, we will be creating our own software-defined network for a small organization. The network, as shown in Figure 1, has four end-hosts and one switch. These devices will run in a virtualized environment on the Computer Science Department's VM server in the Zoo lab, known as SECNET. Each project team will be given their own version of the network to work with and will be able to SSH into each of their hosts to configure the network settings on them. To do so, students must be physically located in the Zoo lab and plug into the isolated network via the red wires. They can then configure their laptop, or ideally a VM running on the student's laptop, to SSH into each of their VMs.

The students will configure each of the four machines to have a unique role. One host will act as a client connecting to a web server. Another will act as a web server distributing content to the client. A third machine will act as a gateway that connects the client and server. The fourth machine will have multiple roles: a DNS server, an OpenFlow controller, and a fake web server (a honey pot). The four machine's triple functionality is simply for convenience (to avoid multiple extra VMs to manage), but the students should envision those functions as being completely independent with no direct interaction between the controller, DNS server, or honeypot web server.

Please note: students **should not change or remove any of the original IPv4 addresses on these VMs**; such changes may result in students being unable to reach their own hosts. However, the students should **add** additional IP alias addresses and address ranges to complete the project. The end-hosts should never communicate with each other using the original IP addresses; all communication between the hosts should be done via the IP aliases.

The SDN will allow the controller to arbitrarily manipulate the traffic flowing between the client and

[1] C.A. Shue, A.J. Kalafut, M. Allman, C.R. Taylor, "On building inexpensive network capabilities," ACM SIGCOMM Computer Communication Review 42 (2), 72-79, 2012.

server and between the DNS server and the client. In particular, the controller will be able to order the gateway to perform NAT operations and elevate traffic to itself in order to inspect and modify it. We will use this to implement the DNS Capabilities approach that we have read and discussed in class.

As an example, consider what happens during a normal web connection request. The operation starts by doing a DNS resolution request. In Figure 2, we show that the DNS request will traverse the virtual switch (provided transparently by the SECNET hypervisor) to reach the Gateway. Using a static rule from the OpenFlow controller, the Gateway will forward the request on to the DNS server.

Once the DNS server receives the request, it will issue a static reply with an IP address for the web server. That response traverses the virtual switch on its way to the client (message 1 in Figure 3). However, the Gateway will not have a rule on how to direct the response back to the client. Instead, it will elevate the entire packet to the OpenFlow controller for consideration (message 2 in Figure 3). The OpenFlow controller will then rewrite the packet's payload to adjust the IP address in the DNS reply to a dynamic address inside the virtual address range for the server. The controller will also keep a copy of this record, showing that the server is associated with the given virtual address with respect to the client. The controller will then send a PACKET_OUT event with this rewritten packet to the Gateway (message 3 in Figure 3). The Gateway will then forward the modified packet on to the client (message 4 in Figure 3).

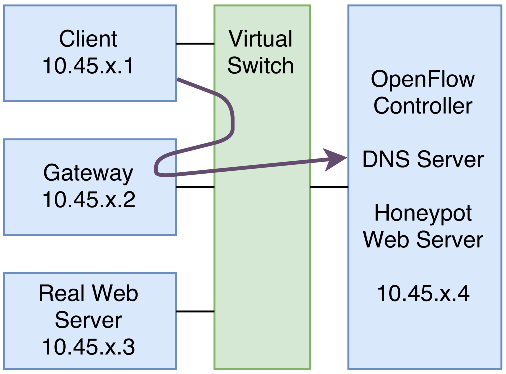


Figure 2: The example packet path for a DNS request from the client.

Once the client receives a DNS reply, it will initiate a web request to the virtual IP address it received in its reply on TCP port 80 (message 1 in Figure 4). This reply will reach the Gateway. The Gateway will not have a rule on how to direct traffic to the given virtual IP address and will thus elevate the entire packet to the controller for review (message 2 in Figure 4). The controller will determine whether the connection should be directed to the real web server or to the honeypot. If it sees a prior DNS response to the client's IP address containing the corresponding virtual IP address within the response's valid DNS time-to-live (TTL), it will instruct the gateway to add a new rule that will send all packets in the connection to the real server using a FLOW_MOD command[2] (message 3 in Figure 4). If the packet did not meet all of those requirements, the controller would instead send a FLOW_MOD that would direct the packet to the honeypot web server instead. The controller must then send the PACKET_OUT message (message 4 in Figure 4) to have the packet delivered to the web server. Importantly, the PACKET_OUT must come after the FLOW_MOD is sent. Finally, the Gateway will deliver the packet to the server (message 5 in Figure 4).

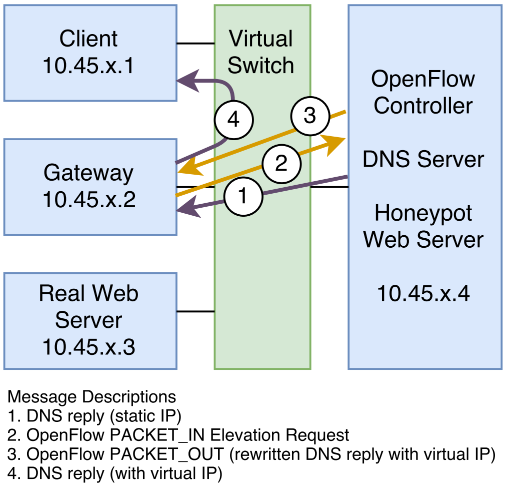


Figure 3: The DNS response from the DNS server is elevated to the OpenFlow controller for inspection and rewriting. Once the OpenFlow controller has rewritten the response, it is returned to the client.

One last problem remains. The Real Web Server and the Honeypot Web Server do not have an IP address alias for every single address in the virtual address space. Instead, they listen on single address in their respective address ranges. Accordingly, the Gateway must translate addresses between the virtual IP address the client uses for the server and the real IP address the server listens on and uses for its replies. Accordingly, the FLOW_MOD rule from the controller must specify this network address translation (NAT) functionality.

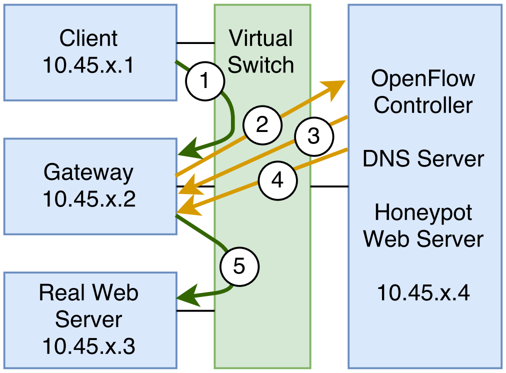
[2] It must also send a FLOW_MOD message to allow the responses from the web server to reach the client, but that is omitted from the diagram for simplicity.

Preliminary Information

Students are asked not to share their code publicly or with students outside of their team, even after the class has ended. This project required a significant WPI infrastructure and time investment to provide a unique and compelling learning opportunity. We would like to reuse some components of the project in future course offerings and want to ensure future WPI students have the same learning opportunities.

The teaching staff will provide each team with access to four Ubuntu 16.04 virtual machines on the Fuller Zoo Lab isolated network. These virtual machines will be available through the `secnet.cs.wpi.edu` VM server. While students will not have access to that server directly, they will have SSH access to the VMs it serves when they are connected through the isolated network in the Zoo Lab. Students can access the network by using the red colored Ethernet cables in that laboratory. Students can obtain credentials and install SSH keys on the VMs via the InstructAssist system at `https://ia.wpi.edu/cs4516/vm_console.php`. Students also have the ability to start, stop, and reboot their VMs through InstructAssist. If a VM becomes unusable due to configuration errors or a failed experiment, students can use the re-image option to restore it to its default state. Be aware that any time the VM is reimaged, **the VM hard drive is reformatted, causing all files on the VM to be permanently lost**. Students are encouraged to develop their code and settings locally and transmit copies to the VMs to avoid data loss.

On some occasions, students may wish to run a packet capture tool, such as Wireshark, to help diagnose network issues. The students should ensure that they have bound the packet capture tool to their wired Ethernet network interface. These precautions will prevent students from accidentally running a packet capture tool on the WPI campus network, which could lead to violations of the campus Acceptable Use Policy (AUP). To avoid negative repercussions, please verify your settings carefully.



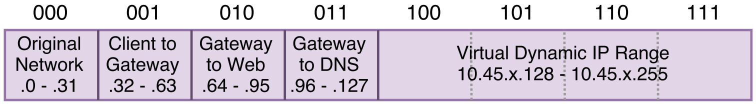
Message Descriptions
1. Web request
2. OpenFlow PACKET_IN Elevation Request
3. OpenFlow FLOW_MOD (authorizes connection via new rule)
4. OpenFlow PACKET_OUT (sends TCP connection)
5. Web request

Figure 4: The web request from the client is elevated to the OpenFlow controller for inspection. If the controller finds it is valid, it adds a new rule authorizing it and sends it back to the Gateway for delivery.

The IP configuration of these hosts will be set by the CS 4516 teaching staff. Students should **not** remove or alter these IP addresses (with `[TEAM_NUM]` representing the student's VM team number):

- **Host 1:** `10.45.[TEAM_NUM].1`
- **Host 2:** `10.45.[TEAM_NUM].2`
- **Host 3:** `10.45.[TEAM_NUM].3`
- **Host 4:** `10.45.[TEAM_NUM].4`

The students should configure their network to uses Host 1 for the client, Host 2 for the Gateway, Host 3 for the Real Web server, and Host 4 for the machine that runs the OpenFlow controller, DNS server, and Honeypot DNS server.



000	001	010	011	100	101	110	111
Original Network .0 - .31	Client to Gateway .32 - .63	Gateway to Web .64 - .95	Gateway to DNS .96 - .127	Virtual Dynamic IP Range 10.45.x.128 - 10.45.x.255			

Figure 5: IP address ranges for the project. The numbers above each square represent the leading three binary digits in the last octet of the address range.

The students will need to add IP aliases to each of these machines. Each team is provided an IP range they may use for this class, which is defined as `10.45.[TEAM_NUM].0/24` in CIDR notation (or `10.45.[TEAM_NUM].*` in wildcard notation). In Figure 5, we show how students should subdivide their address ranges. Students should assume a CIDR prefix length of 27. In other words, the IP address range is divided into 8 pieces using the first three leading binary digits of the last octet of the IP address. For example, the last IP octet for the first group ranges from `0000 0000` to `0001 1111`, which expresses the range from `10.45.[TEAM_NUM].0` to `10.45.[TEAM_NUM].31` in dotted decimal. That range is specifically reserved for the initial IP addresses assigned to the four hosts. The second group's last octet ranges from `0010 0000` to `0011 1111`, which expresses the range from `10.45.[TEAM_NUM].32` to `10.45.[TEAM_NUM].63` in dotted decimal. For convenience, we treat the last four of the eight /27's as a single group that represents the virtual IP address range. Students can think of that CIDR notation as `10.45.[TEAM_NUM].128/25`.

For consistency, we recommend the students use the following IP network configurations for their hosts:

Host Number	Original Address	Additional Alias(es)	Additional Static Routes
1	10.45.[TEAM_NUM].1	10.45.[TEAM_NUM].33/27	10.45.[TEAM_NUM].128/25 via gateway 10.45.[TEAM_NUM].34
2	10.45.[TEAM_NUM].2	10.45.[TEAM_NUM].34/27 10.45.[TEAM_NUM].65/27 10.45.[TEAM_NUM].97/27	
3	10.45.[TEAM_NUM].3	10.45.[TEAM_NUM].66/27	10.45.[TEAM_NUM].0/25 via gateway 10.45.[TEAM_NUM].65
4	10.45.[TEAM_NUM].4	10.45.[TEAM_NUM].98/27	

Deliverable Deadlines and Point Values

The deadlines for each of the phases are specified in the course syllabus. The point values for each of the four phases are specified in the section heading. At a high level, Phase 1 is allotted for one week and is worth 25 points. Phases 2, 3, and 4 are allotted twelve to fourteen days and are weighted 50, 50, and 35 points respectively.

Phase 1: Basic Network Infrastructure and Testing (25 points)

In the preliminary phase, the class will create the basic network, web server, and DNS server infrastructure that will be used throughout the term. The students should configure the IP addresses and routes on the hosts as described in the Preliminary Information section above. The students should also configure a Web server on Hosts 3 and 4 using the Apache2 web server. The web servers should be loaded with an initial index.html file with a simple message that indicates whether the server is the real web server or the honeypot.

To create the DNS server, students should install the Berkeley Internet Name Domain, the most widely used DNS server on the Internet. Ubuntu users can install this using the `bind9` package. The DNS zone file should have an `A` entry for the web server in the format `www.team[TEAM_NUM].4516.cs.wpi.edu`. For example, the zone file entry for Team 1 would be:

```
www.team1.4516.cs.wpi.edu.  A 10.45.1.10
```

While the 10.45.1.10 server does not actually exist, when the controller is fully implemented, it will rewrite the IP address to a virtual IP address anyway, so the client will never see this address in practice. Students should use their VMs to test connectivity with the other VMs on their newly configured IP aliases. Note that IP aliases can be quite confusing; since each VM now has multiple addresses, the students will need to keep track of the source IP address used for network probes in addition to the destination IP address.

Finally, the students should install Open vSwitch on Host 3 and the appropriate OpenFlow controller software on Host 4. Students may choose to use either the Pox or the Floodlight OpenFlow controller software in this project. The students should explore both projects and determine the best fit for the project. Students are encouraged to pick the best tool for the job rather than simply picking the tool based on the programming language that the tool uses.

In completing this phase, the students should turn in the following:

1. A transcript of all the relevant commands they ran on each machine in separate files. Use the file naming schema of `transcript_host[NUMBER].txt` to organize your files. Students can use the `history` command to obtain a transcript, copy it into a text file, and then edit out the lines containing irrelevant commands.

2. The output of the `ifconfig` and `route -n` commands on each of the machines using the file naming schema `outputs_host[NUMBER].txt`.

3. A document, named `justification.txt`, that provides a justification for why the students picked the OpenFlow controller they selected. The justification should include a strengths and limitations list for both Floodlight and Pox.

Phase 2: Basic OpenFlow Functionality (50 points)

In this phase, students will test their basic OpenFlow functionality between the Gateway and Controller. The students should add an IP alias on the web server for one IP address within the virtual IP address space temporarily to enable this phase of testing. The students will then create an OpenFlow controller application that approves all DNS and TCP traffic to reach the destination without modification or packet inspection. The students will demonstrate that the functionality is correct with proper traffic flow by providing packet captures at the hosts and explaining their observations.

The students will then compare the performance of a traditional network with an OpenFlow network. The students will build tools and scripts that allow them to test the connection performance of a traditional web request in their network when OpenFlow is not in use (the Gateway simply employs IPv4 forwarding) verses when OpenFlow asks a controller for permission (even when that controller approves everything). The students will perform 100 trials in both scenarios and report their findings. Using strategically placed packet captures and clock synchronization, the students will collect a small number of sample interactions and perform timing analysis to determine where most of the time is spent between the client's initial TCP `SYN` packet and the client receiving the TCP `SYN+ACK` response.

In completing this phase, the students should turn in the following:

1. The configuration files for the Open vSwitch software on the Gateway.

2. The module code for the OpenFlow controller that approves all the traffic.

3. Packet captures demonstrating the proper operation with an accompanying explanation of those captures.

4. Any custom written testing tools or scripts with a document explaining the students' testing procedure.

5. A document comparing the results in the 100 trials and discussion about whether this analysis is expected.

6. A timing analysis on the sample flows indicating what latency is added by each component during the initial TCP round trip. This should include packet captures that validate the timing analysis.

Phase 3: Enabling Address Translation and DNS Processing (50 points)

The students will remove the temporary virtual IP alias in Phase 2. They will then enable the appropriate NAT and DNS operations to enable the controller to appropriately map virtual IPs to the correct real Web server and the honeypot server. The system will successfully parse and rewrite DNS packets and correlate virtual IP addresses in new TCP flows with the records stored from prior DNS packet parsing. The controller will appropriately direct traffic to the honeypot server when the records do not match (or when expired) and will otherwise direct the client to the real web server.

The students will create an appropriate testing harness that allows them to verify the proper operation over 100 trials. The students will verify that the controller properly expires exceptions for new flows when the DNS TTL expires.

In completing this phase, the students should turn in the following:

1. The module code for the OpenFlow controller that performs the necessary operations, including the DNS packet parsing, the packet rewriting, and the NAT authorizations.
2. Any custom written testing tools or scripts with a document explaining the students' testing procedure.
3. Any supporting evidence, such as packet captures, that definitively demonstrate the system works appropriately.

Phase 4: OpenFlow on End-hosts (35 points)

The students will now add Open vSwitch to the client machine and create an IP alias on the controller to allow it to directly communicate with the client. When the client creates a new TCP connection, it will now seek permission from the controller to originate the traffic. At that point, the controller will issue the appropriate `FLOW_MOD` and `PACKET_IN` messages that will cause the client to rewrite the source address of all of the packets it originates to be a random IP address selected from the `10.45.[TEAM_NUM].48/28` address range. The controller will also order the client to create the appropriate rules to rewrite incoming packets destined to that address back to `10.45.[TEAM_NUM].33`.

The students will evaluate the approach's performance and compare it with the results from Phase 2. The students will then analyze whether the results are consistent with the results from Phase 2 or if they appear to deviate. Finally, the students should determine the advantages and disadvantages of this approach.

In completing this phase, the students should turn in the following:

1. The configuration files for the Open vSwitch software on the client.
2. The code for the OpenFlow controller module that performs the appropriate rule manipulations on the client.
3. A document comparing and contrasting the performance of the approach with the results from Phase 2.
4. Finally, the students should answer the following in a document named `answers.txt`:
   (a) Does the approach offer any security benefits? If so, what are they? Are there any other benefits that could result from using the system?
   (b) The IPNL paper described a strong concerns about network address translation. Do these same concerns apply to the NAT approach implemented in Phase 4? Why or why not?
   (c) What implications would this end-host OpenFlow/NAT approach have on a network like WPI's? When two machines on the LAN communicate, what would we expect to happen? What about when these machines communicate with off LAN systems?

Resources and Restrictions

When programming for the controller, all students are required to use only the programming language associated with the controller. The controller application must be wholly contained within the controller's code itself. Students may use existing tools built into the controllers, but will need prior approval from the teaching staff before using any outside libraries. No credit will be given to solutions that do not adhere to these requirements, so students should contact the teaching staff for any clarification before using outside resources.

As always, we encourage students to avail themselves to Internet resources and Linux manual pages when completing the assignment. Socket tutorials, such as `http://beej.us/guide/bgnet/output/html/singlepage/bgnet.html`, will be helpful in understanding socket programming. However, each student must ensure that he or she writes his or her own code and explicitly mentions any resources used **including the linked web tutorial, materials provided in class, and discussions with other students outside the team**. In case of any questions about making use of a specific Web resource for the project, seek clarification in advance.

Assignment Submission and Deliverables

The following instructions apply to each of the phases. For each file, students should put their name at the top (including in the comments of source code files). Each phase specifies the files that should be submitted. All of the files to be submitted should be placed in a directory. The directory should then be compressed into a .zip archive that is 30 MBytes in size or less. This archive should be uploaded to InstructAssist (`https://ia.wpi.edu/cs4516/files.php`) in the appropriate project phase folder. Please only submit your work in standard zip archives to simplify the grading process. Formats such as .tar, .rar, .gz, .7z, .bz2, and .xz are not acceptable.

All designs, documentation, and source code should be in plain text or PDF format. Please do not submit proprietary document formats, such as .doc or .docx.

The file submission is only a portion of the deliverables for most project phases. Students should use the following checklist in turning in their projects to avoid forgetting any deliverables:

1. Sign up for a project partner or have one assigned (URL: `https://ia.wpi.edu/cs4516/request_teammate.php`),
2. Submit the project code and documentation via InstructAssist (URL: `https://ia.wpi.edu/cs4516/files.php`),
3. Complete a Partner Evaluation (URL: `https://ia.wpi.edu/cs4516/evals.php`), and
4. Schedule a Project Demonstration (URL: `https://ia.wpi.edu/cs4516/demos.php`), which may be posted slightly after the submission deadline.

The remainder of this document is devoted to rubrics for each of the phases of the project. These rubrics will be used for grading purposes and uploaded with student scores. Students are encouraged to review these rubrics to ensure they understand how their grades will be determined.

Project Rubric: Phase 1: Basic Infrastructure and Testing

				Partner Evaluation?
Grader:	____________	Student Name:	________________________	________
Date/Time:	____________	Student Name:	________________________	________
Team ID:	____________	Student Name:	________________________	________
Late?:	____________			

Project Score: / 25

Earned	**Points**	**Task ID**	**Description**
_____	8	0	Transcripts of commands from all four hosts. The commands must include a record of the key alterations performed at each host.
_____	7	1	Appropriate ifconfig and route –n output for each of the hosts.
_____	10	2	Proper justification of the selected DNS controller, including strengths and limitations of each.

Grader Notes:

Project Rubric: Phase 2: Basic OpenFlow Functionality

				Partner Evaluation?
Grader:	________________	Student Name:	______________________________	__________
Date/Time:	________________	Student Name:	______________________________	__________
Team ID:	________________	Student Name:	______________________________	__________
Late?:	________________			

Project Score: / 50

Earned	**Points**	**Task ID**	**Description**
_____	5	0	Correct configuration files for Open vSwitch software on Gateway.
_____	15	1	OpenFlow module code that correctly approves all traffic.
_____	10	2	Packet captures that show proper operation with accompanying explanations. Prerequisite: Task 1.
_____	10	3	Custom tools and scripts with testing procedure explanation. Prerequisite: Task 1.
_____	10	4	Comparison and analysis of 100 trials. Prerequisite: Task 1.
_____	10	5	Timing analysis and packet captures of sample network flows. Prerequisite: Task 1.

Grader Notes:

Project Rubric: Phase 3: NAT and DNS

				Partner Evaluation?
Grader:	________________	Student Name:	______________________________	__________
Date/Time:	________________	Student Name:	______________________________	__________
Team ID:	________________	Student Name:	______________________________	__________
Late?:	________________			

Project Score: / 50

Earned	Points	Task ID	Description
_____	30	0	OpenFlow module code that correctly performs DNS parsing, packet rewriting, and NAT operations.
_____	10	1	Testing tools and scripts with document explaining testing procedure. Prerequisite: Task 0.
_____	10	2	Supporting evidence of proper operation. Prerequisite: Task 0.

Grader Notes:

Project Rubric: Phase 4: OpenFlow on End-Hosts

				Partner Evaluation?
Grader:		Student Name:		
Date/Time:		Student Name:		
Team ID:		Student Name:		
Late?:				

Project Score: / 35

Earned	**Points**	**Task ID**	**Description**
_____	5	0	Correct configuration files for Open vSwitch software on client.
_____	15	1	OpenFlow module code that correctly applies appropriate rules on the client.
_____	8	2	Performance results for Phase 4 and comparison and contrast with Phase 2 results. Prerequisite: Task 1.
_____	7	3	Thorough and correct response contained in the answers.txt document. Prerequisite: Task 1.

Grader Notes: