Recall that the ARP protocol typically maintains a cache of IP-to-Ethernet address translation pairs on your computer. The `arp` command (in both MSDOS and Linux/Unix) is used to view and manipulate the contents of this cache.\(^1\)

We begin by examining the contents of the ARP cache on your computer. On my Mac, you can check the contents of the ARP cache by typing `arp -a`. On other Linux/Unix machines it may take some hunting. The executable for the `arp` command can be in various places. Popular locations are `/sbin/arp` (for linux) and `/usr/etc/arp` (for some Unix variants). Check out the MAN pages for the `arp` command on your system.

**Exercise 1 [2]: ARP caching** Run the `arp` command and write down the contents of your computers ARP cache. What is the meaning of each column value?

In order to observe your computer sending and receiving ARP messages, we’ll need to clear the ARP cache, since otherwise your computer is likely to find a needed IP-Ethernet address translation pair in its cache and consequently not need to send out an ARP message. Clearing the ARP cache on the my Mac by typing `sudo arp -d -a` and then entering your administrative password.\(^2\)

**Exercise 2 [4]: ARP in action** Open Firefox and make sure its cache is empty. Type the following URL into your browser, but do not hit the enter key just yet.

---

\(^1\)Since the `arp` command and the ARP protocol have the same name, it’s understandably easy to confuse them. But keep in mind that they are different - the `arp` command is used to view and manipulate the ARP cache contents, while the ARP protocol defines the format and meaning of the messages sent and received, and defines the actions taken on message transmission and receipt.

\(^2\)In order to run this command you’ll need root privileges. If you don’t have root privileges and can’t run Wireshark on a Windows machine, you can skip the trace collection part of this lab and just use `wireshark-arp-trace.pcapng`. 
Open a terminal window and start Wireshark. Now, you clear the ARP cache as above. Check that the cache is empty and then immediately return to the browser and hit the return key. The reason for the rush is that the cache refills pretty quickly, defeating the purpose of this exercise.

Your browser should once again display the rather lengthy US Bill of Rights. Stop Wireshark packet capture. Again, we’re not interested in IP or higher-layer protocols, so change Wireshark’s “listing of captured packets” window so that it shows information only about protocols below IP. You should now see an Wireshark window that looks like:

In the example above, frame 25 contains my first ARP request messages asking who is heck is 149.130.178.1. The reply comes from the horse’s mouth and tells me the router’s ethernet address.

*a* [2] What are the hexadecimal values for the source and destination addresses in the Ethernet frame containing the ARP request message?

*b* [2] Give the hexadecimal value for the two-byte Ethernet Frame type field. What upper layer protocol does this correspond to?

**Exercise 3 [4]: ARP specifications** Download the ARP specification using the *ftp* protocol in your browser window:

---

3It’s our first hop router, but since I cleared my ARP cache, I forgot.

A readable, detailed discussion of ARP is also at

Answer the following questions.

a [1] How many bytes from the very beginning of the Ethernet frame does the ARP opcode field begin?

b [1] What is the value of the opcode field within the ARP-payload part of the Ethernet frame in which an ARP request is made?

c [2] Does the ARP message contain the IP address of the sender?

**Exercise 4 [6]: ARP reply message** Now find the ARP reply that was sent in response to the ARP request.

a [1] How many bytes from the very beginning of the Ethernet frame does the ARP opcode field begin?

b [1] What is the value of the opcode field within the ARP-payload part of the Ethernet frame in which an ARP response is made?

c [2] Where in the ARP message does the answer to the earlier ARP request appear the IP address of the machine having the Ethernet address whose corresponding IP address is being queried?

d [2] What are the hexadecimal values for the source and destination addresses in the Ethernet frame containing the ARP reply message?
Problems

Problem 1 [10]: Wireless  Answer each of the following questions about the link layer. Briefly justify your answers.

a [2]: Beacon frame  Describe the role of the beacon frames in 802.11

b [2]: RTS/CTS frames  True or false (Briefly justify your answer): Before an 802.11 station transmits a data frame, it must first send an RTS frame and receive a corresponding CTS frame.

c [2]: RTS threshold  Describe how the RTS threshold works.

d [2]: Moving day  Suppose a wireless station moves from one BSS to another within the same subnet. When the APs are interconnected with a switch, an AP may need to send a frame with a spoofed MAC address to get the switch to forward the frame properly. Why?

e [2]: Mobil nodes  If a node has a wireless connection to the Internet, does that node have to be mobile? Explain. Suppose that a user with a laptop walks around her house with her laptop, and always accesses the Internet through the same access point. Is this user mobile from a network standpoint?

Problem 2 [6]: WiFi Access  Suppose there are two ISPs providing WiFi access in a particular cafe, with each ISP operating its own AP and having its own IP address block.

a [3]: Channel 11 collisions  Further suppose that by accident, each ISP has configured its AP to operate over channel 11. Will the 802.11 protocol completely break down in this situation? Discuss what happens when two stations, each associated with a different ISP, attempt to transmit at the same time.

b [3]: Different channels  Now suppose that one AP operates over channel 1 and the other over channel 11. How do your answers change?

Problem 3 [10]: Hidden nodes  Consider the scenario shown in Figure 7.34 of the text, in which there are four wireless nodes, A, B, C, and D. The radio coverage of the four nodes is shown via the shaded ovals; all nodes share the same frequency. When A transmits, it can only be heard/received
by B; when B transmits, both A and C can hear/receive from B; when C
transmits, both B and D can hear/receive from C; when D transmits, only
C can hear/receive from D.

Suppose now that each node has an infinite supply of message that it
wants to send to each of the other nodes. If a message’s destination is not
an immediate neighbor, then the message must be relayed. For example, if
A wants to send to D, a message from A must first be sent to B, when then
sends the message to C, which then sends the message to D. Time is slotted,
with a message transmission time taking exactly one time slot, e.g., as in
slotted Aloha. During a slot a node can do one of the following: (i) send
a message; (ii) receive a message (if example one message is begin sent to
it), (iii) remain silent. As always, if a node hears two or more simultaneous
transmissions, a collision occurs and none of the transmitted messages are
received successfully. You can assume here that there are no bit-errors, and
thus if exactly one message is sent, it will be received correctly by those
within the transmission radius of the sender.

a [2] Suppose that an omniscient controller (i.e, a controller that knows
the state of every node in the network) can command each node to what-
ever it (the omniscient controller) wishes, i.e., to send a message, to re-
ceive a message, or to remain silent. Given this omniscient controller,
what is the maximum rate at which a data message can be transferred
from C to A, given that there are no other message between any other
source/destination pairs?

b [2] Suppose that A sends message to B, and D Sends messages to C.
What is the combined maximum rate at which data message can flow from
A to B and from D to C?

c [2] Suppose that A sends message to B, and C sends message to D.
What is the combined maximum rate at which data messages can flow
from A to B and from C to D?

d [2] Suppose that the wireless links are replace by wired links. Repeat
questions (a) – (c) above for this scenario. Solution.

e [2] Finally suppose that we are again in the wireless scenario, and that
for every data message sent from source to destination, the destination
will send an ACK message back to the source (e.g., as in TCP). Suppose
also that each ACK message takes up one slot. Repeat questions (a) – (c)
avove for this scenario.
Problem 4 [4]: Authentication Protocols  Consider the authentication protocol in Figure 8.18 (6th Edition) of the text in which Alice authenticates herself to Bob, which we saw works well (i.e., we found no flaws in it). Now suppose that while Alice is authenticating herself to Bob, Bob must authenticate himself to Alice. Give a scenario by which Trudy, pretending to Alice, can now authenticate herself to Bob as Alice. (Hint: Consider that the sequence of operations of the protocol, one with Trudy initiating and one with Bob initiating, can be arbitrary interleaved. Pay particular attention to the fact that both Bob and Alice will use a nonce, and that if care is not taken, the same nonce can be used maliciously.

Problem 5 [4]: Textbook RSA  Using RSA, choose \( p = 3 \) and \( q = 11 \), and encode the word “dog” by encrypting each letter separately. Each letter is assigned a number according to its position in the alphabet. Hint: Nine is not a bad choice for \( e \). You will need to calculate \( d \) from \( p, q, \) and \( e \). Apply the decryption algorithm the encrypted version to recover the original plaintext message. Incidentally, the encryption technique we just performed goes by another name. What is it and does this give you pause about the security of textbook RSA?

Problem 6 [6]: Diffie-Hellman Exchange  The Diffie-Hellman (DH) algorithm allows two entities to agree on a shared key over an unsecured channel. It makes use of a large prime number \( p \) and another large number \( g \) less than \( p \). Both \( p \) and \( g \) are made public (so that an attacker would know them). In DH, Alice and Bob each independently choose secret keys, \( S_A \) and \( S_B \), respectively. Alice then computes her public key, \( T_A \), by raising \( g \) to \( S_A \) and then taking mod \( p \). Bob similarly computes his own public key \( T_B \) by raising \( g \) to \( S_B \) and then taking mod \( p \). Alice and Bob then exchange their public keys over the Internet. Alice then calculates the shared secret key \( S \) by raising \( T_B \) to \( S_A \) and then taking mod \( p \). Similarly, Bob calculates the shared key \( S' \) by raising \( T_A \) to \( S_B \) and then taking mod \( p \).

a [2] Argue that, in general, Alice and Bob obtain the same symmetric key; that is, prove \( S = S' \).

b [2] With \( p = 11 \) and \( g = 2 \), suppose Alice and Bob choose private keys, \( S_A = 5 \) and \( S_B = 12 \), respectively. Calculate Alice’s and Bob’s public keys, \( T_A \) and \( T_B \). Show all work.

c [2] Following up on the previous subproblem, calculate \( S \) as the shared symmetric key. Show all work.