

Question 1. Each time you type a line while using your IM client, that line is sent as a TOC packet stored within the data field of an Ethernet frame.¹ Each character in the TOC packet is encoded in ASCII using 8 bits or 1 byte. One line you have probably typed at some point is “Hi”.

- (a) Assuming that you sent this message to “thommurtagh” and that the only sources of overhead involved are the remaining fields of the TOC packet and the Ethernet frame used to carry this packet through the Ethernet, what percentage of the frame is useful (i.e. used to represent the two characters in the actual message “Hi”)? Assume that the packet is formatted using the modern Ethernet packet layout (shown in class) rather than the experimental Ethernet packet (shown in the paper by Metcalfe and Boggs). Justify your answer.
- (b) What percentage of the frame is overhead (i.e., fields required by the protocols in order to transport your data)? Justify your answer.

Question 2. Suppose that five computers named A, B, C, D, and E are waiting for a transmission to finish on an Ethernet. All transmit at once when the previous packet is finished and collide.

Table ?? on the last page shows a partial history of how these five computers might contend to obtain access to the network after this collision. In constructing this table, we have deliberately ignored variability in collision detection times so that if two computers choose the same random number when deciding to delay their next attempt to transmit, then they will begin their next transmissions at exactly the same time. We have also assumed the amount of time consumed by a collision is exactly equal to the time a computer will pause before transmitting if it randomly decides to wait for one slot.

In the table, we indicate which of computers A through D attempted to transmit in each time slot by showing the range of back-off times (e.g., “0..3”) from which each computer that is involved in a collision must choose its next back-off time and the random back-off value it actually selected (e.g., “:2”). We left the entries in the table for computer E empty.

Fill in the column for computer E in a way that minimizes the number of times E attempts to transmit under the assumption that the contention process does not end until round 8 (when computer C transmits alone). Show which slots E transmits in by indicating the range of back-off values it must choose between and the back-off value it selects after each collisions in which it is involved. There are several possible solutions. You need only provide one.

Question 3. When a network tries to deliver a packet, the efficiency of the process can be measured by dividing the time spent actually sending the bits of a packet by the total time from when the system starts trying to send the packet to the point when the last bit is sent. If no time is wasted, the efficiency will be 1. If something delays the beginning of the transmission of the packet, the efficiency will be:

$$\text{time_spent_sending} / (\text{length_of_delay} + \text{time_spent_sending})$$

Obviously, if the delay is greater than 0, the efficiency will be less than 1. In Ethernet, the main cause of inefficient transmission is collisions. If a computer encounters no collisions when it sends a message, there is no delay and the efficiency is 1. However, if a computer encounters one or more collisions before it manages to send a packet successfully, then the time spent resolving the collisions is a non-zero delay and the efficiency will be less than one. Note that in this description we talk about the time spent sending a packet rather than the time spent delivering a packet. That is, we are only considering the time required to transmit the packet. This does not include the time required for the packet to travel through the cable to its destination.

Suppose that on some Ethernet, two computers, A and B, both have a packet to send. Assume that A starts sending before B, but not long enough before B to avoid a collision. Also assume that after detecting the collision, A attempts to send its packet again as soon as it detects that the network is again idle and that nothing collides with this second attempt allowing A to deliver its packet successfully. For the following questions assume the network is 1 kilometer long, computers A and B are at opposite ends of the network, that the packet is 1000 bits long and the data rate is 10 million bits per second and that signals can travel at the speed of light (3×10^8 meters/sec)

- (a) What is the time spent sending the packet? Provide both a formula and a numeric answer to this and parts (b) and (c).

¹ Actually, each TOC packet is placed within a FLAP packet which is placed within a TCP packet which is placed within an IP packet which is placed in the data field of the Ethernet frame, but we will ignore this for now.

- (b) What is the delay caused by a single collision?
- (c) What is the efficiency if exactly one collision occurs before A transmits successfully?
- (d) Now, assume that the network in question is 2 kilometers long rather than 1 kilometer but that all other factors described above remain the same (including that A and B are at opposite ends of the network). What would be the efficiency associated with A's transmission to B? In general, as the distance between A and B increases would the efficiency of transmissions increase or decrease?
- (e) Suppose instead that the network is as described above (including being 1 kilometer long) but that its data transmission rate was increased to 100 million bits per second. What would be the efficiency associated with A's transmission to B? In general, as the transmission rate increases would the efficiency of transmissions increase or decrease?

Question 4. Consider the following two classes `Gossip` and `GossipChain`:

```
public class Gossip {

    // The target of the gossip
    private String target;

    // The gossip itself
    private String scuttlebutt;

    public Gossip(String target, String scuttlebutt) {
        this.target = target;
        this.scuttlebutt = scuttlebutt;
    }

    public String getTarget() {
        return target;
    }
}

public class GossipChain {

    private Gossip current;
    private GossipChain next;
    private boolean empty;

    public GossipChain() {
        empty = true;
    }

    public GossipChain(Gossip current, GossipChain next) {
        empty = false;
        this.current = current;
        this.next = next;
    }

    public GossipChain next() {
        return next;
    }

    public boolean end() {
        return empty;
    }
}
```

```

    }

    public Gossip getGossip() {
        return current;
    }
}

```

The `GossipChain` class is an example of a linked list. Each `GossipChain` object has a `Gossip` object as well as a link to the next chain of gossip. The end of a `GossipChain` is denoted by an empty `GossipChain` object. In particular, these objects have an empty member variable which is `true`. Suppose we also had the following function `giveMeAChain` which returns a chain of gossip:

```

public GossipChain giveMeAChain() {
    GossipChain gc = new GossipChain();
    int i = 0;
    while (i < 5) {
        String s = "" + i;
        gc = new GossipChain(new Gossip(s, "Did you hear about " + s + "?"), gc);
        i = i + 1;
    }
    return gc;
}

```

Suppose that we create a new chain of gossip:

```
GossipChain chain = giveMeAChain();
```

Now consider the following questions.

- What does `chain.next().next().getGossip().getTarget()` return?
- We say the length of `chain` is 5 because there are 5 non-empty items of gossip in the chain. Write a method for `GossipChain` called `length` with signature `public int length()` that returns the length of a given gossip chain. For example, `chain.length()` would return 5 and `new GossipChain().length()` would return 0.
- Suppose a `GossipChain` object has n items of gossip. Write a method for the `GossipChain` class called `half` with signature `public GossipChain half()` that returns the `GossipChain` at position $\lfloor n/2 \rfloor$ where $\lfloor i \rfloor$ rounds i down to the nearest integer. For example `chain.half().getGossip().getTarget()` would return 2.

Table 1: The Backoff Table

SLOT	A	B	C	D	E
0	backoff range 0..1: delay used: 0	backoff range 0..1: delay used: 1	backoff range 0..1: delay used: 0	backoff range 0..1: delay used: 0	
1	backoff range 0..3: delay used: 3		backoff range 0..3: delay used: 3	backoff range 0..3: delay used: 2	
2		backoff range 0..3: delay used: 1			
3					
4		backoff range 0..7: delay used: 6		backoff range 0..7: delay used: 6	
5	backoff range 0..7: delay used: 1		backoff range 0..7: delay used: 2		
6					
7	backoff range 0..15: delay used: 4				
8			C transmits alone!		