

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.

The TOC packet would contain the type ‘‘toc2_send_im’’, the buddy name ‘‘thommurtagh’’, two spaces and two quotes in addition to the message. This is a total of 29 characters requiring 29 bytes. This data would then be placed in an Ethernet frame which (according to the slide of the Ethernet packet format presented in class and through the course web site) includes 12 bytes of addresses, a 4 byte error check, a 2 byte length/type field and an 8 byte preamble for a total frame size of $8+12+2+29+4 = 55$ bytes.

In reality, however, on an Ethernet there is a minimum required packet size designed to ensure that a station always transmits long enough to detect any collision that might occur. As a result the minimum data field size is 46. This means that to send this TOC packet in an Ethernet frame, a computer would have to add 17 bytes of 0s or other data to fill the 46 byte data field. The resulting packet would contain 72 bytes. Of these 72 bytes, 2 are actual data. That is, 2.8% of the packet is useful data.

- (b) What percentage of the frame is overhead (i.e., fields required by the protocols in order to transport your data)? Justify your answer.

97.2% of the packet described in part (a) is overhead.

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.

Three possible solutions are shown in the tables at the end of this document.

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

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.

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).

The time required to send a packet is simply the size of the packet (in bits) divided by the rate at which bits are transmitted (in bits per second). In the notation of the paper, this is P/C . Given the values described in the problem this would be $\frac{1000\text{bits}}{10^7\text{bits/second}}$ or 10^{-4} seconds (i.e. 100 microseconds).

- (b) What is the delay caused by a single collision?

Given the scenario described in the problem, the delay lasts from when A begins to transmit its packet for the first time until the beginning of A's second, successful transmission attempt. A will not start its second attempt until it can detect that B has stop transmitting. Therefore, the delay is equal to the time from when A starts to transmit until the last bit of B's interrupted transmission travels past A.

Even though B starts after A, B will continue transmitting until the first bit of A's message reaches B. The time required for this to happen is the time it takes the signal to travel along the full length of the network. This is just the length of the network divided by the speed of light = $\frac{1000\text{meters}}{3 \times 10^8\text{m/sec}} = \frac{1}{3} \times 10^{-5}$ seconds. B will abort its transmission at this point, but A will not realize that this has happened until the bit B was sending as it detected A's message travels down the network in the opposite direction to A. This will take another $\frac{1}{3} \times 10^{-5}$ seconds. Therefore, the total delay will be $\frac{2}{3} \times 10^{-5}$ seconds.

- (c) What is the efficiency if exactly one collision occurs before A transmits successfully?

The efficiency is just the time actually spent sending a packet divided by the total time required to send it (i.e. the sending time plus the delay caused by contention). In this scenario the efficiency is therefore:

$$\frac{10^{-4}\text{seconds sending time}}{(10^{-4}\text{seconds sending time} + \frac{2}{3} \times 10^{-5}\text{delay time})} = \frac{1}{(1 + \frac{2}{3} \times 10^{-1})} = \frac{1}{1.0666} = 94\%$$

- (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?

Increasing the length of the network will double the delay time giving an efficiency of :

$$\frac{1}{(1 + \frac{4}{3} \times 10^{-1})} = 88\%$$

In general, as the size of the network increases, its efficiency will 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?

This change would decrease the actual sending time by a factor of 10 leaving an efficiency of:

$$\frac{.1}{(.1 + \frac{2}{3} \times 10^{-1})} = \frac{1}{(1 + \frac{2}{3})} = 60\%$$

In general, as the transmission rate increases, the efficiency will 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.

(a) What does `chain.next().next().getGossip().getTarget()` return?

"2"

(b) 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.

```

public int length() {
    if ( empty ) {
        return 0;
    } else {
        return 1 + next.length();
    }
}

```

- (c) 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.

One simple way to accomplish this is to write a helper method that removes a prefix of a specified length from a chain:

```
private GossipChain removePrefixOfLength( int n ) {
    if ( n == 0 ) {
        return this;
    } else {
        return next.removePrefixOfLength( n - 1 );
    }
}

public GossipChain half() {
    return this.removePrefixOfLength( this.length() / 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	<i>backoff range 0..1: delay used: 0</i>
1	backoff range 0..3: delay used: 3		backoff range 0..3: delay used: 3	backoff range 0..3: delay used: 2	<i>backoff range 0..3: delay used: 1</i>
2		backoff range 0..3: delay used: 1			<i>backoff range 0..7: delay used: 4</i>
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				<i>backoff range 0..15: delay used: 4</i>
8			C transmits alone!		

Table 2: 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	<i>backoff range 0..1: delay used: 1</i>
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			<i>backoff range 0..3: delay used: 1</i>
3					
4		backoff range 0..7: delay used: 6		backoff range 0..7: delay used: 6	<i>backoff range 0..7: delay used: 2</i>
5	backoff range 0..7: delay used: 1		backoff range 0..7: delay used: 2		
6					
7	backoff range 0..15: delay used: 4				<i>backoff range 0..15: delay used: 4</i>
8			C transmits alone!		

Table 3: 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	<i>backoff range 0..1: delay used: 1</i>
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			<i>backoff range 0..3: delay used: 2</i>
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		<i>backoff range 0..7: delay used: 1</i>
6					
7	backoff range 0..15: delay used: 4				<i>backoff range 0..15: delay used: 4</i>
8			C transmits alone!		