COMP40660 ASSIGNMENT 1

YIQIU LEI 16206801

Usage:

This program have two parts. The first part is a constructor (all.java). This constructor contains two double values. One is the normal total time to send 1500 bytes and another one is the best total time to send 1500 bytes.

Run the main.java then enter the protocol, standard and data rate according the prompts. Then the calculation results will be shown in the console. If the users want to get another calculation, just print “y” or “Y” in the console after the question “Do you want to do the next calculation?”

If the user entered a wrong protocol, standard or data rate, there is a prompt in the console. If you enter a wrong protocol, you can also print “y” or “Y” in the console after the question “Do you want to do the next calculation?” to get another calculation. If you enter a wrong standard or a wrong data rate, you can re-run this program to get another calculation.

Calculation:

Math.ceil() is used to get a higher integer value from a decimal. For example: Math.ceil(3.79)=4
Math.floor() is used to get a lower integer value from a decimal. For example: Math.ceil(3.79)=3

DIFS = SIFS + 2*Slot;
Number of bits per symbol = (NBits * CRate * NChan) * Nss

Time to transmit data for best:

```
Data_best =
    Symbol_duration*Math.ceil((((datasymbol+MAC_header+SNAP_LLC_header)*8)+tail)/Math.
floor(nbits*crate*nchan_best));
```

Time to transmit data for normal:

```
Data_normal =
    Symbol_duration*Math.ceil((((datasymbol+MAC_header+SNAP_LLC_header)*8)+tail)/Math.
floor(nbits*crate*nchan_normal));
```

To calculate the symbols of RTS, CTS, ACK:

```
RTS_normal=Math.ceil((20*8+6)/Math.floor(nbits*crate*nchan_normal));
CTS_normal=Math.ceil((14*8+6)/Math.floor(nbits*crate*nchan_normal));
```

ACK_normal=Math.ceil((14*8+6)/Math.floor(nbits*crate*nchan_normal));
RTS_best=Math.ceil((20*8+6)/Math.floor(nbits*crate*nchan_best));
CTS_best=Math.ceil((14*8+6)/Math.floor(nbits*crate*nchan_best));
ACK_best=Math.ceil((14*8+6)/Math.floor(nbits*crate*nchan_best));

Just 802.11g has signal extension, so for other standard, signal extension = 0.

Total time to transmit 1500 bytes by UDP:

all_normal = DIFS + Preamble_normal + RTS_normal * Symbol_duration + SIFS + Preamble_normal
+ CTS_normal * Symbol_duration + SIFS+ Preamble_normal + Data_normal + SIFS
+ Preamble_normal + ACK_normal * Symbol_duration + 4 * signal_exe;
all_best = DIFS + Preamble_best + RTS_normal * Symbol_duration + SIFS + Preamble_best
+ CTS_normal * Symbol_duration + SIFS+ Preamble_best + Data_best + SIFS + Preamble_best
+ ACK_normal * Symbol_duration + 4 * signal_exe;

Total time to transmit 1500 bytes by TCP:

all_normal= DIFS + Preamble_normal+ RTS_normal * Symbol_duration+ SIFS + Preamble_normal
+ CTS_normal * Symbol_duration + SIFS+ Preamble_normal + Data_normal + SIFS
+ Preamble_normal + ACK_normal * Symbol_duration + DIFS + Preamble_normal
+ RTS_normal * Symbol_duration + SIFS + Preamble_normal + CTS_normal * Symbol_duration
+ SIFS + Preamble_normal + Symbol_duration * Math.ceil((((TCP_ACK + MAC_header
+ SNAP_LLC_header) * 8) + tail)/ Math.floor(nbits * crate * nchan_normal * Nss_normal))
+ SIFS + Preamble_normal + ACK_normal * Symbol_duration + 8 * signal_exe;
all_best = DIFS + Preamble_best + RTS_best * Symbol_duration + SIFS + Preamble_best
+ CTS_best * Symbol_duration + SIFS+ Preamble_best + Data_best + SIFS + Preamble_best
+ ACK_best * Symbol_duration + DIFS + Preamble_best+ RTS_best * Symbol_duration + SIFS
+ Preamble_best + CTS_best * Symbol_duration + SIFS + Preamble_best + Symbol_duration *
Math.ceil((((TCP_ACK + MAC_header + SNAP_LLC_header) * 8) + tail)/ Math.floor(nbits *
crate * nchan_best * Nss_best))+ SIFS + Preamble_best + ACK_best * Symbol_duration
+ 8 * signal_exe;

Assume back off (BO) = 0, and there are not collisions.

Throughput = (1500 bytes * 8bits/byte)/ total time to send 1500 bytes.
The amount of time needed to transfer 10 GB of data =
10*1024*1024*1024 bytes/((1500/total time to send 1500 bytes)*1000000);

Why there is a difference between the actual throughput and the advertised data rate.

When there are some data which will be transmitted, there are some information such as headers to ensure that the data will be transmitted to the correct destination.

If there will be a collision, the collision will be prevented. That is because there are many idle

periods such as DIFS, SIFS etc. When there is a transmission, other transmissions will be paused. There are also some frames such as RTS, CTS, ACK to ensure the transmission to be successful.

All of these are important but will influence the transmission, so there are differences between actual data rate and advertised data rate.

802.11 performance improves after each release. Briefly discuss the trade-offs involved in such improvements.

- 802.11a's operating frequency is 5GHz. The maximum raw data transfer rate is 54Mbit/s, which meets the requirement of medium throughput (20Mbit/s) of the real network. Since 2.4GHz band is already widely used, the adoption of 5GHz band gives 802.11a the advantage of less conflict. However, people who use 2.4GHz is not able to use this.
- 802.11g changes the carrier frequency to 2.4GHz so that many users who use 2.4GHz have a better feeling.
- 802.11n improves network throughput and reliability. Increase channel bandwidth (20MHz-40MHz). Use MIMO (Multiple Input/Multiple Output). It works in both 2.4GHz and 5GHz bands.
- 802.11ac is much faster (Speeds of up to 1Gbps (Theoretical)). Wider channels 80MHz or 160MHz. MU-MIMO (Multiple User-MIMO) - enables multiple users to simultaneously access the same channel. However, it only operates in the 5GHz band so that many users who use 2.4GHz cannot use. Use 256QAM to modulate.
- 802.11ax improves network throughput which is nearly 4 times of 802.11ac's network throughput and reliability. Use 1024QAM to modulate. Use MU-MIMO and OFDMA.