ITS 413 – ASSIGNMENT 2 ANSWERS

First name: ______________________________ Last name: ___________________________

ID: _____________________________ Total Marks: ____________
out of 70

Due Date: Friday 16 February 2007, 9am (you can hand in before the start of the lecture)

I certify that, unless otherwise acknowledged, all work carried out in this assignment is my own.

Sign Name: ________________________________ Date: ________________

Guidelines

1. This is an individual assignment. You are not allowed to work with other students, including checking your answers and providing help (if someone wants help, they should ask me!).
2. The assignment must be typed on a computer. It should be handed in on A4 sheets with a single staple in the top left corner. No plastic sleeves etc.
3. You must attach this Cover Sheet (including name, ID and signature) to the front of your assignment.
4. I only accept hardcopies of the assignment – that is, I will not accept an emailed copy of assignment.
5. Show all your working out and calculations. If you get the answer wrong, you may still receive marks for your calculations. If you only provide an answer (and no calculations), then you will be penalised.
6. State and explain any assumptions that you make.
7. When you are asked to discuss or explain something, then it is important that your explanation is very clear to the reader (me). Ambiguous or unclear explanations may receive 0 marks.
8. There are many resources on the Internet. You should of course make use of them, but do not restrict your study to just one resource (e.g. Wikipedia). You should use multiple sources, and acknowledge all sources in your references. Also make sure you write in your own words and draw your own diagrams.

Question 1 [20 marks]

Write pseudo-code that describes the implementation of the sender's congestion control algorithm for TCP described in RFC2581.

You must define all parameters that you use. You can make assumptions, but you should explain or at least make a comment that an assumption has been made. For example, RFC2581 allows you to use different approaches to implement the algorithm – make sure you indicate which approach you have chosen.

Hint1: It is common to structure a protocol implementation in the following way:

```
while(1)
    if EventA then
        if ConditionA then
            ActionA
        else if ConditionB then
            ActionB
    else if Event B then
        Action C
    end if
end while
```

You should use a similar structure.

Hint2: The events you need to consider are:

1. Receive an expected ACK
2. Retransmission Timeout
3. Receive 3rd Duplicate ACK
4. Receive ACK for Retransmitted Packet

In practice, there may be other events (such as Receive 4th Duplicate ACK) – for simplicity, you can ignore these special cases.

Hint3: You only need to show how the congestion control parameters change. You can ignore actions such as setting timers and sending packets.

Hint4: Section 3 of RFC2581 contains the main details of the algorithm.

```
/* SMSS: Sender Maximum Segment Size */
/* FlightSize: Amount of outstanding data in network */
/* Initial value must be less than or equal to 2*SMSS and no more than 2
segments */
cwnd = 2*SMSS
ssthresh = rwnd /* ssthresh can be any arbitrary value */
While (1)
      If Receive Expected ACK Then
            /* This part is optional */
            If In Fast Recovery Phase Then
                  Cwnd = ssthresh
                  Stop Fast Recovery Phase
            Else
                  If cwnd > ssthresh Then /* Congestion Avoidance */
                        /* Lets assume the approximation of Eq. 2 */
                        cwnd += SMSS*SMSS/cwnd
                        /* Note the limitation of the approximation */
                        If cwnd = 0 Then
                              cwnd = 1
                        End If
                  Else /* Slow Start */
                        cwnd += SMSS
                  End If
            End If
      Else If Retransmission Timeout Then
            ssthresh = Max(FlightSize/2, 2*SMSS)
            cwnd = 1*SegmentSize /* or Loss Window */
      Else If Receive 3rd Duplicate ACK Then
            ssthresh = Max(FlightSize/2, 2*SMSS)
            cwnd = ssthresh + 3*SMSS
            Start Fast Recovery Phase
      Else If Receive ACK for Retransmitted Packet Then
            Cwnd += SMSS
      End If
End While
```

Question 2 [30 marks]

a) Complete the design of the Gnutella peer-to-peer protocol.

   The lecture notes provide an informal (and incomplete) description of the Gnutella protocol, including the 5 message types and some of the operations. You must provide a complete design of the protocol. You should follow the 5 components of a protocol. For the protocol rules, I suggest you use pseudo-code (see Question 1 for example) or state tables (a state-machine could also be used, but may not be the easiest approach!).

   You do not need to define all fields in the messages, nor their sizes. But you should at least list the most important fields that are needed to make the protocol work.

   You can assume there are no firewalls, and ignore the Push message (and any related functionality).

   Note that as the protocol description in the lecture notes is incomplete, you will need to choose how to design the remaining parts of the protocol. You should explain why you make certain design decisions.

b) The network on the last page shows the connections for a Gnutella network (e.g. the Ping/Pong have completed, and C=3). Assume node 1 issues a search for a file stored on node 17. Nodes do not use a cache of search results. Once the query reaches node 17, it will give a unidirectional response along the same path the query arrived on. You can assume that each transmission takes the same time, t_h. Use your protocol design to answer the following:

   i. What is the number of messages sent using the Gnutella protocol?

   ii. How long does it take for node 1 to receive the first response?

   iii. Answer parts (i) and (ii), but assuming the file is stored on node 20, not node 17.

   iv. Answer parts (i) and (ii), but assuming the file is stored on both node 17 and node 20.

c) Part (a) uses the basic Gnutella protocol. Explain how the expanding ring search and random walker algorithms work for Gnutella, and then compare all three algorithms. You should use the figure on the next page to illustrate how the two algorithms work. (You may make copies of the figure, or download the Visio or Word file).

Five Elements of Protocol: Service; Assumptions; Vocabulary; Encoding; Protocol Rules

Service

Gnutella is an application layer protocol that uses TCP as a transport layer. It provides a service to a user (or application).

Assumptions

- Transport layer provides a reliable, ordered, end-to-end delivery of packets
- A new node knows the IP addresses of initial peers

Messages

Ping

- Source node ID (this is in fact optional – you may use the IP address/port number to uniquely identify a node, instead of this ID. The IP/port are included in the IP/TCP packet headers)
- Total size of file shared (this is useful information for a new node to tell other nodes what amount of data it shares)

Pong

- Source node ID
- Total size of files shared

Query

- Source Node ID
- Search criteria – can be a string or regular expression
- TTL – probably between 0 and 255
- Query ID – a unique ID to identify this query, e.g. 16-bit integer

QueryHit

- Source Node ID
- List of resources/files found. May optionally include information that also describes each resource (such as file size, keywords)
- Query ID

```
While (1)
      If User Submits Query(q) Then
            /* We may store a local cache of responses and use that */
            /* where relevant. I do not show any code to maintain */
            /* that cache (e.g. delete old entries) */
            If q is in local cache Then
                  Send Response(r) to User
            Else
                  id = Generate Query ID
                  Send query(n,q,ttl,id) to peers
                  Start Response Timer(id)
            End If
      Else If Receive query(n,q,ttl,id) Then
            /* We should only process a query once */
            If id not already received Then
                  If q is in local cache Then
                        Send queryhit(n,r,id) to n
                  Else If ttl-1 > 0 Then
                        Send query(n,q,ttl-1,id) to peers
                  End If
            End If
      Else If Receive queryhit(n,r,id) Then
            If n == current Node Then
                  If Response Timer(id) > 0 Then
                        Send Response(r) to User
                        Stop Response Timer(id)
                  End
                  Save q in local cache
            Else
                  Send queryhit(n,r,id) to n
            End If
      Else If Timeout Response Timer(id) Then
            Send Error to User
      Else If User Joins Network Then
            Send ping(n,s) to peers
            Start Ping Timer
      Else If Receive ping(n) Then
            /* We may not want to respond to a ping (we may not */
            /* want to become a peer). The decision to respond or */
            /* not would be within some local policy */
            If within policy Then
                  Send pong(n) to n
            End If
            /* If we are broadcasting pings we need to forward it */
            /* In fact, would need additional code, similar to that */
            /* used by query to maintain TTL for pings. This is not */
            /* shown here. */
            If broadcast ping Then
                  Send ping(n) to peers
            End If
      Else If Receive pong(n) Then
            Add source node to list of possible peers
            Update list of C permanent peers
      Else If Timeout Ping Timer Then
            /* When we join, we should create our list of */
            /* permanent peers based on all pong responses */
            /* received within a certain time */
            Update list of C permanent peers
      End If

End While
```

Part (b)

Lets assume it takes Th seconds to send a message to neighbours. If we assume that in each “round” of sending that if two nodes are scheduled to send to each other then in fact one will send first and then the other will not send.

The number of messages can be counted from the diagram where in round 1 node 1 sends to nodes 2, 3 and 4 and so on. Hence there are 3 messages in round 1, 6 in round 2, then 8, 8 and 5 in round 5. A total of 30 messages. Of course the replay involves 4 transmissions, therefore the answer is 34 messages.

But if we assumed that if two nodes did transmit at the same time, then there will be an additional 9 messages. Answer would be 43.

Delay is (4+4)* Th = 8^{Th} since the request takes 4 hops: 1-2-11-17.

If the result is on node 20, not 17, then the only difference is the result takes 3 hops. Hence answer is 33 messages and delay is 6Th

If the result is on both 17 and 20 then delay is 6Th and number of messages is 34 + 3 = 37 since both 20 and 17 respond.

Part (c)

Expanding Ring: Instead of sending a query with TTL=Max (e.g. Max = 7), the node first sends a query with TTL=1. Then if no response is received within a certain timeout period, then the query is sent again but with TTL=2, and the process repeats until TTL=Max. The advantage of this is that once the result is found, then no more queries are forwarded. That is, if the result was 3 hops away, then effectively the number of messages sent is equivalent to broadcasting with TTL=3. As opposed to normal operation where the number of messages sent is equivalent to broadcasting with TTL=7 (Max), even if the result is 3 hops away. The disadvantage is that there will be additional delay in getting the result because the initiator must wait before sending the second (and subsequent) query.

Random Walk: Initiator randomly selects a subset of permanent peers and sends to them (rather than all peers). And the peers do the same. This reduces the number of messages sent (not broadcast to all nodes) but also reduces the chance of finding the result quickly (i.e. increases the delay). In general, the TTL must be set higher than the normal mode in order to find the same set of results.

Question 3 [20 marks]

Suggest ways in which the SIIT website can be optimised for search engines. You should divide your methods into “good” and “bad” – where the “bad” methods are those that use questionable (either ethically or legally) techniques. You should explain why you think these methods may be questionable. You can focus on the following pages of the SIIT website (but may give recommendations applicable to the entire site):

- http://www.siit.tu.ac.th/indexs.html
- http://www.siit.tu.ac.th/overview/intro.html
- http://www.siit.tu.ac.th/program/undergrad.html

Hint: You should try to describe the methods to enough detail such they could be presented to the Web site committee in the SIIT Computer Centre.

Note that SIIT website usually comes up first when searching for SIIT on Google, however on search for more generic phrases (e.g. "international technology university Thailand" and similar) it does not rank as well.

Marking: This question will be marked based on the accuracy/appropriateness of the methods suggested, as well as the number of methods and a clear description of what the Computer Centre should do to implement it. Also, I will rank the answers from all students – for example, the best answers will be given the most marks, while others will be marked relative to this.

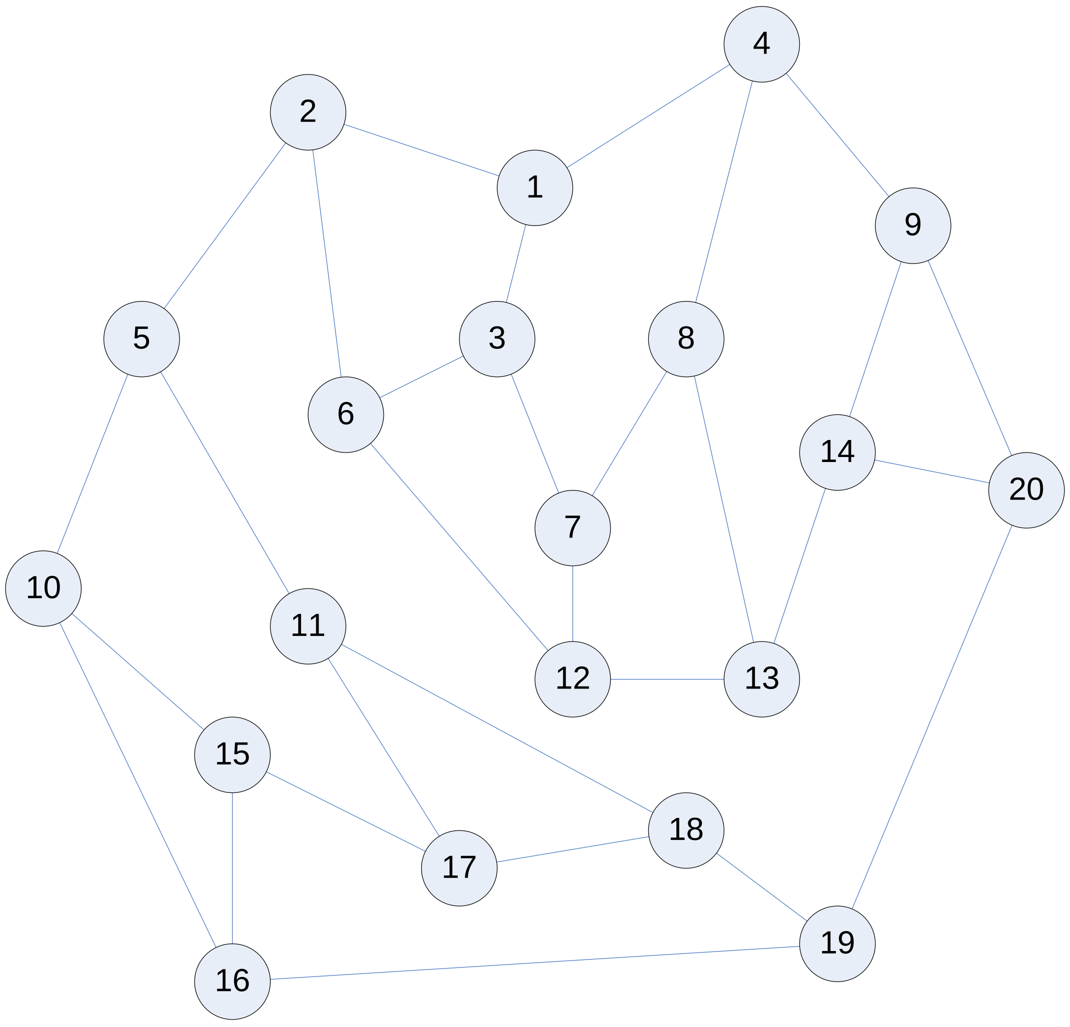
Some general optimisation techniques:

- *Expand the title: e.g. "Sirindhorn International Institute of Technology, the leading English-based centre for Engineering and Science university education in Thailand"*
    - *This introduces several new keywords*
- *Similar to above for H1 tag*
- *Use H2 and H3 tags, and include relevant information*
- *Submit SIIT to all major search engines and directories*
- *Edit public wikis on general topics and universities to appropriately reflect SIIT (e.g. Wikipedia, national and international lists of Universities)*
- *Ensure Thammasat (and other Thammasat centre) websites prominently link to SIIT*
- *Discuss with Keidanren and FTI about obtaining links from their web pages to SIIT. Similar for any other organisations associated with SIIT (Ministry of Education, Toshiba Thailand, SIIT.net, …)*
- *Include more descriptive text on the front page (e.g. an introductory paragraph saying what SIIT is, and what it does – relevant keywords should be used early)*
- *Most of the above comments also are relevant for the intro.html and undergrad.html pages. In fact these two pages do not use H1 or H2 fields, so these is a definite improvement that is needed.*
- *Provide links to other relevant sites throughout the text. As a result, more people may visit those sites, and the benefits of a link exchange with the other site increase.*
- *Could potentially rename pages with more meaning for names:*
    - *Intro -> education_and_research_at_SIIT*
    - *Undergrad -> English_based_undergraduate_engineering_at_SIIT*

  Then links to these pages from other sites may rank higher

All the general "bad" techniques that were described in lecture notes could of course be relevant.

Gnutella network for Question 2


1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20