Networked Communications Design Project

Context and Objectives

The objective of this design project is to design a client (or clients) which can reliably recover the content of a message by snooping individual packets from a server. There are many important constraints on the snooping process, and it is important to understand that up to three snoopers with different IP addresses (different computers here) can be used together to recover the message.

The arrangement is sketched in Figure 1, but note that there are many details not shown on the figure. One item not shown is an HTTP server to which one of your computers should connect, in order to send the deciphered message once you have it. You will need to implement an HTTP client, using TCP as the underlying transport, to post your message and receive a success or failure notification. Once successful, the message server selects a new message automatically, which you should again proceed to try to recover. In this way, your designed solution is expected to recover messages sequentially, earning points for each message recovered, until you have either recovered all messages or a timer expires.

Your main objective is to recover all messages before the timer expires, and to do so with a maximum score, where the score is reduced if you receive failure codes in response to your HTTP posts. A secondary objective is to recover all the messages as fast as possible – i.e., well before the timer expires, if possible.

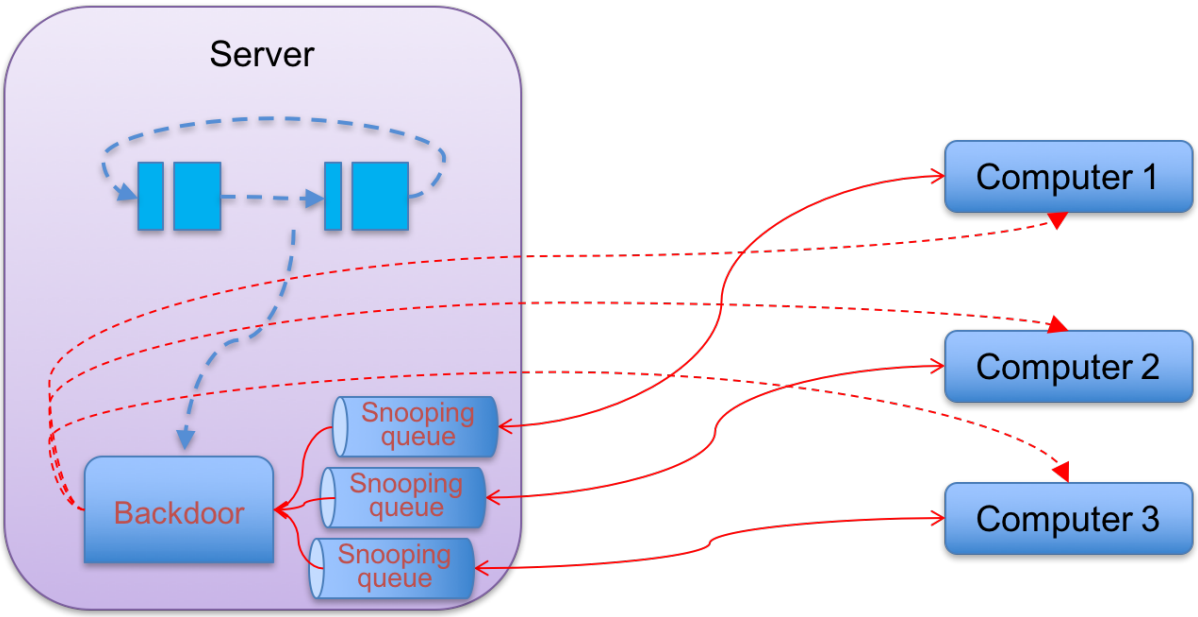

Figure 1: Snoop-me system overview

Details of the expiry time **T**, message rate **R** and typical number of total message characters **C** will be provided on the course Moodle page by Monday of Week 9.

Detailed Description of the Snoop-Me System

1. The server runs in a loop, repeating a current message over and over again until your system has successfully recovered it.
    a. Notionally, the server is sending the message repeatedly to a client somewhere else, and you are "snooping" the communication, but the existence of this other client is not important to you.
    b. The message consists of a sequence of ASCII characters in the range 0x01 to 0x7F, including special characters. The special character ctrl-D (i.e., 0x04) marks the end of the message and will not occur elsewhere.
    c. The message is partitioned into packets of variable length, with the shortest packets containing as little as 4 characters and the longest packet having up to 20 characters. The packetization of the message does not change over its repetitions.
    d. Each message packet has an associated 8-byte big-endian 64-bit integer identifier that increments sequentially, with natural wrap-around. The identifier does not repeat with the message, and its initial value is essentially random, at least as far as you are concerned.
    e. The message characters are transmitted at a constant rate of **R** characters per second, which means that message packets are transmitted at a non-constant rate, since they have variable length. The data rate does not include any allowance for the packet identifier, so the interval between the start of a packet p with C_p characters and the start of the next packet is $C_p/\mathbf{R}$ seconds.
2. The server provides a "backdoor" with three (3) snooping channels.
    a. Each snooping channel has an associated queue, whose length **L** is initially 2, meaning that it can hold up to two snooping requests.
    b. Snooping requests are sent to a designated port on the server as UDP datagrams. The port assigned to your team will be given out in the first lab in Week 9.
    c. Each snooping request datagram r shall be 8 bytes long, representing two unsigned big-endian 32-bit integers S_r and P_r, in that order.
    d. Before sending each packet, the server checks each snooping queue for a request r. If one is found, its S_r value is decremented by 1. If this results in S_r becoming 0, the request is removed from the queue and a copy of the packet is sent to the snooper as a UDP datagram, except that the first 4 bytes of the 8-byte packet identifier are replaced by P_r. That is, the UDP datagram that is sent to the snooper commences with the request datagram's 4-byte P_r field, followed by the last 4 bytes (32 LSBs) of the message identifier, and then the message characters themselves. Be very careful not to send a snooping request with S_r=0 here, since S_r will need to be decremented 2^{32} times before this produces 0.
    e. The three snooping channels are independent, but they get assigned to the snooper's IP address when a snooping request first arrives. When the server is in the "exclusive" mode, used for final testing, no IP address may be assigned to multiple snooping channels, so you will need to exploit three separate computers to use the full set of snooping channels.

3. Snooping too often is dangerous. The server keeps track of the time between snooping responses that it has sent – not the time between requests it has received -- where time is measured in characters, since characters are transmitted at the constant rate **R**. If this inter-response time drops below 50 characters, you have been "**exposed**." Each time you are exposed, the relevant snooping channel's queue length is reduced by 1, until it reaches 0. The queue length is auto-incremented by 1, up to a maximum length of 2, every 1000 characters. Thus, being exposed is expected to really slow you down and perhaps interfere with the behaviour of your algorithm. You do not receive any notification that you have been exposed.
4. Once you have recovered the message, you should post a copy to the HTTP server at the port number that will be supplied to your team in the first laboratory in Week 9.
    a. You will need to make sure you understand the syntax for a correct HTTP POST request – not an HTTP GET request.
    b. You will receive back either an HTTP 200 series (success) response code or an HTTP 406 response status code (not acceptable). In the former case the server moves on to the next message. In the later case, you need to keep trying but your final score for the message is scaled by 0.9. If you receive two failures for the same message your score for that message is scaled twice by 0.9 (i.e. 0.81) and so forth.
    c. You are to use the HTTP/1.1 protocol with a persistent TCP connection from exactly one of the IP addresses that is involved in snooping. You may establish the connection at any time, but may not reconnect until the server is restarted.
    d. Once the final message is successfully recovered, the HTTP response code will be 205, rather than 200. You can then disconnect, gracefully or otherwise from the HTTP server. Any further HTTP requests will result in an HTTP response status code 404.
    e. Other HTTP 400-series response status codes may be issued if you construct your requests improperly.

Constraints

Your design should ideally be implemented in C or C++ and must use BSD sockets. In particular, it is expected that your program(s) use the "select" function to wait for I/O completion, rather than less portable methods such as Event signaling or callback functions that are offered in the Windows sockets model. The BSD model is available on all operating systems of interest.

If you wish to use a different programming language than C or C++, you should explain your intent and rationale to the course convenor at the start of Week 9 in order to get final permission.

It is perfectly allowable for the three client computers to communicate with one another, using any desired protocol, so long as such communication is implemented using C or C++ or another language authorized by the course convener.

You may use your own computers, if desired, rather than the computers in the lab, but see the discussion below regarding physical and online participation.

Physical and Online Participation in this Design Project

The Snoop-Me server will be hosted on a virtual machine local to UNSW. You should be able to communicate directly with the server via a VPN, but the preferred way to communicate with the server as an online participant is to SSH into your team account on a local machine – details will be provided at the start of Week 9.

It is expected that online students maintain close interaction with their team members throughout the laboratory via Microsoft Teams. You are strongly recommended to do this using a laptop or one of the laboratory computers with a shared Teams session, rather than using phones.

Design Outcome (Elective Performance) Mark

The primary requirement for this project is recovery of the messages. Your message recovery score is given by

$$M = \frac{25}{N} \sum_{n=1}^{N} 0.9^{F_n}$$

where N is the total number of messages that the server had to send, whether you recovered them all or not, and F_n is the number of HTTP failures reported for message n. The score M has a maximum value of 25.

The secondary objective for the project is to recover messages as quickly as possible – ideally well before the timeout $\boldsymbol{T}$. This message recovery speed score is a linear function of the degree to which your design completes the recovery of all messages prior to $\boldsymbol{T}$, reach a maximum of 10 marks if you recover all messages in the time $\boldsymbol{T}_{\mathbf{min}}$, which will be specified on the course Moodle page no later than Monday of Week 9. If you do not recover all messages within the timeout period $\boldsymbol{T}$, your message recovery speed score must be 0.

The value of **T** will be much larger than $\mathbf{T}_{\mathbf{min}}$ to emphasize the importance of message recovery, as opposed to speed. In particular, since message failure is penalized in the recovery score, you should prefer designs that minimize the risk of error over designs that rush to a quick conclusion.

Overall, your design outcome mark is worth 10% of your assessment for the course. The mark will be awarded out of 50 as follows:

- Message recovery score M: (____/25)
- Message recovery speed (full marks for recovery in $\boldsymbol{T}_{\mathbf{min}}$ seconds): (____/10)
- Design principles, elegance, etc. (must recover at least 1 message): (____/10)
- Software structure and clarity of code (must recover at least 1 message): (____/5)

Note: The values **R**, **C**, **T** and $\boldsymbol{T}_{\mathbf{min}}$ will be provided by Monday of Week 9. In the event that your design recovers all messages in less than $\boldsymbol{T}_{\mathbf{min}}$ seconds, bonus marks may possibly be awarded to compensate for less than perfect performance in the last two areas listed above.

Understanding Marks

You will be assessed individually on Friday of Week 10 regarding your understanding of your team's design, including your contribution to the design. This understanding mark is worth 5% of the overall course assessment and is awarded out of 25, as follows:

- Understanding of system design, trade-offs and choices: (____/10)
- Understanding of individual sub-systems designed: (____/10)
- Understanding of how things could be further improved: (____/5)

Note: the first two items mentioned above must be supported by a **draft report provided in hard copy form** in the laboratory on Friday of Week 10. The draft report must contain at least a close approximation to the final algorithm. Informal notes can provide any remaining details by the time of marking. The draft report **must also identify the individual sections** written by (or that will be written by) each team member.