CNP3-Practice Documentation

Release 0.0

Olivier Bonaventure Mickael Hoerdt, Laurent Vanbever and Virg

October 07, 2010

CONTENTS

This document contains the questions for the practical part of the INGI2141 course during the 2010-2011 academic year. The html and pdf versions will be updated every week. These exercises have been written by Olivier Bonaventure with the help of Mickael Hoerdt, Damien Saucez and Laurent Vanbever and Virginie van den Schriek

CHAPTER

ONE

THE ALTERNATING BIT PROTOCOL

The objective of this set of exercises is to better understand the basic mechanisms of the alternating bit protocol and the utilisation of the socket interface with the connectionless transport service.

1. Consider the Alternating Bit Protocol as described in the book.

- How does the protocol recover from the loss of a data segment ?
- How does the protocol recovers from the loss of an acknowledgement ?

2. A student proposed to optimise the Alternating Bit Protocol by adding a negative acknowledgment, i.e. the receiver sends a *NAK* control segment when it receives a corrupted data segment. What kind of information should be placed in this control segment and how should the sender react when receiving such a *NAK* ?

3. Transport protocols rely on different types of checksums to verify whether segments have been affected by transmission errors. The most frequently used checksums are :

   - the Internet checksum used by UDP, TCP and other Internet protocols which is defined in **RFC 1071** and implemented in various modules, e.g. http://ilab.cs.byu.edu/cs460/code/ftp/ichecksum.py for a python implementation
   - the 16 bits or the 32 bits Cyclical Redundancy Checks (CRC) that are often used on disks, in zip archives and in datalink layer protocols. See http://docs.python.org/library/binascii.html for a python module that contains the 32 bits CRC
   - the Alder checksum defined in **RFC 2920** for the SCTP protocol but replaced by a CRC later
   - the Fletcher checksum

   By using your knowledge of the Internet checksum, can you find a transmission error that will not be detected by the Internet checksum ?

4. The CRCs are efficient error detection codes that are able to detect :

   - all errors that affect an odd number of bits
   - all errors that affect a sequence of bits which is shorter than the length of the CRC

   Carry experiments with one implementation of CRC-32 to verify that this is indeed the case.

5. Checksums and CRCs should not be confused with secure hash functions such as MD5 defined in **RFC 1321** or SHA-1 described in **RFC 4634**. Secure hash functions are used to ensure that files or sometimes packets/segments have not been modified. Secure hash functions aim at detecting malicious changes while checksums and CRCs only detect random transmission errors. Perform some experiments with hash functions such as those defined in the http://docs.python.org/library/hashlib.html python hashlib module to verify that this is indeed the case.

6. A version of the Alternating Bit Protocol supporting variable length segments uses a header that contains the following fields :

   - a number (0 or 1)
   - a length field that indicates the length of the data

- a CRC

To speedup the transmission of the segments, a student proposes to compute the CRC over the data part of the segment but not over the header. What do you think of this optimisation ?

7. On Unix hosts, the `ping(8)` command can be used to measure the round-trip-time to send and receive packets from a remote host. Use `ping(8)` to measure the round-trip to a remote host. Chose a remote destination which is far from your current location, e.g. a small web server in a distant country. There are implementations of ping in various languages, see e.g. http://pypi.python.org/pypi/ping/0.2 for a python implementation of ping
8. How would you set the retransmission timer if you were implementing the Alternating Bit Protocol to exchange files with a server such as the one that you measured above ?
9. What are the factors that affect the performance of the Alternating Bit Protocol ?
10. Links are often considered as symmetrical, i.e. they offer the same bandwidth in both directions. Symmetrical links are widely used in Local Area Networks and in the core of the Internet, but there are many asymmetrical link technologies. The most common example are the various types of ADSL and CATV technologies. Consider an implementation of the Alternating Bit Protocol that is used between two hosts that are directly connected by using an asymmetric link. Assume that a host is sending segments containing 10 bytes of control information and 90 bytes of data and that the acknowledgements are 10 bytes long. If the round-trip-time is negligible, what is the minimum bandwidth required on the return link to ensure that the transmission of acknowledgements is not a bottleneck ?
11. Derive a mathematical expression that provides the *goodput* achieved by the Alternating Bit Protocol assuming that :
    - Each segment contains D bytes of data and c bytes of control information
    - Each acknowledgement contains c bytes of control information
    - The bandwidth of the two directions of the link is set to B bits per second
    - The delay between the two hosts is s seconds in both directions

    The goodput is defined as the amount of SDUs (measured in bytes) that is successfully transferred during a period of time
12. The socket interface allows you to use the UDP protocol on a Unix host. UDP provides a connectionless unreliable service that in theory allows you to send SDUs of up to 64 KBytes.

- Implement a small UDP client and a small UDP server (in python, you can start from the example provided in http://docs.python.org/library/socket.html but you can also use C or java)
- run the client and the servers on different workstations to determine experimentally the largest SDU that is supported by your language and OS. If possible, use different languages and Operating Systems in each group.

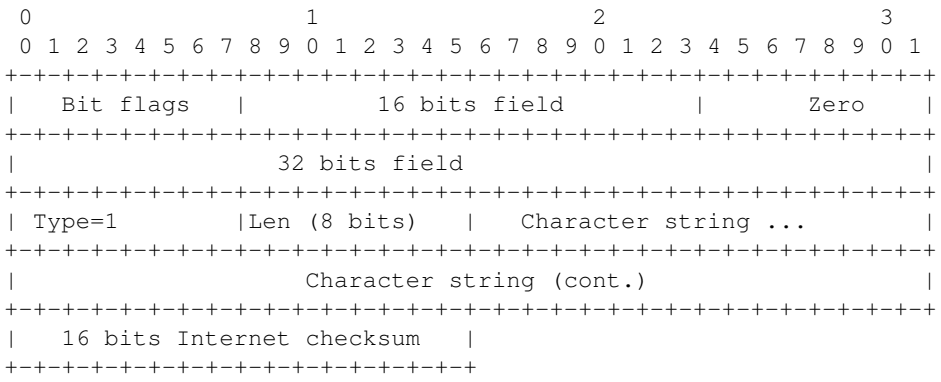
13. By using the socket interface, implement on top of the connectionless unreliable service provided by UDP a simple client that sends the following message shown in the figure below.

    In this message, the bit flags should be set to *01010011b*, the value of the 16 bits field must be the square root of the value contained in the 32 bits field, the character string must be an ASCII representation (without any trailing *0*) of the number contained in the 32 bits character field. The last 16 bits of the message contain an Internet checksum that has been computed over the entire message.

    Upon reception of a message, the server verifies that :
    - the flag has the correct value
    - the 32 bits integer is the square of the 16 bits integer
    - the character string is an ASCII representation of the 32 bits integer
    - the Internet checksum is correct

    If the verification succeeds, the server returns a SDU containing *11111111b*. Otherwise it returns *01010101b*

Inside each group, implement two different clients and two different servers (both using different langages). The clients and the servers must run on both the Linux workstations and the Sun server (*sirius*). Verify the interoperability of the clients and the servers inside the group. You can use C, Java or python to write these implementations.



14. Consider an Alternating Bit Protocol that is used over a link that suffers from deterministic errors. When the error ratio is set to $\frac{1}{p}$, this means that $p - 1$ bits are transmitted correctly and the p^{th} bit is corrupted. Discuss the factors that affect the performance of the Alternating Bit Protocol over such a link.

CHAPTER

TWO

GO-BACK-N AND SELECTIVE REPEAT

Go-back-n and selective repeat are the basic mechanisms used in reliable window-based transport-layer protocols. These questions cover these two mechanisms in details. You do not need to upload the time sequence diagrams on the svn repository, bring them with you on paper.

1. Amazon provides the S3 storage service where companies and researchers can store lots of information and perform computations on the stored information. Amazon allows users to send files through the Internet, but also by sending hard-disks. Assume that a 1 Terabyte hard-disk can be delivered within 24 hours to Amazon by courier service. What is the minimum bandwidth required to match the bandwidth of this courier service ?
2. Several large datacenters operators (e.g. Microsoft and google) have announced that they install servers as containers with each container hosting up to 2000 servers. Assuming a container with 2000 servers and each storing 500 GBytes of data, what is the time required to move all the data stored in one container over one 10 Gbps link ? What is the bandwidth of a truck that needs 10 hours to move one container from one datacenter to another.
3. What are the techniques used by a go-back-n sender to recover from :

- transmission errors
- losses of data segments
- losses of acknowledgements

4. Consider a b bits per second link between two hosts that has a propagation delay of t seconds. Derive a formula that computes the time elapsed between the transmission of the first bit of a d bytes segment from a sending host and the reception of the last bit of this segment on the receiving host.
5. Consider a go-back-n sender and a go-back receiver that are directly connected with a 10 Mbps link that has a propagation delay of 100 milliseconds. Assume that the retransmission timer is set to three seconds. If the window has a length of 4 segments, draw a time-sequence diagram showing the transmission of 10 segments (each segment contains 10000 bits):

- when there are no losses
- when the third and seventh segments are lost
- when the second, fourth, sixth, eighth, ... acknowledgements are lost
- when the third and fourth data segments are reordered (i.e. the fourth arrives before the third)

6. Same question when using selective repeat instead of go-back-n. Note that the answer is not necessarily the same.
7. Consider two high-end servers connected back-to-back by using a 10 Gbps interface. If the delay between the two servers is one millisecond, what is the throughput that can be achieved by a transport protocol that is using 10,000 bits segments and a window of

- one segment

- ten segments
- hundred segments

8. Consider two servers are directly connected by using a *b* bits per second link with a round-trip-time of *r* seconds. The two servers are using a transport protocol that sends segments containing *s* bytes and acknowledgements composed of *a* bytes. Can you derive a formula that computes the smallest window (measured in segments) that is required to ensure that the servers will be able to completely utilise the link ?
9. Same question as above if the two servers are connected through an asymmetrical link that transmits *bu* bits per second in the direction used to send data segments and *bd* bits per second in the direction used to send acknowledgements.
10. The Trivial File Transfer Protocol is a very simple file transfer protocol that is often used by diskless hosts when booting from a server. Read the TFTP specification in **RFC 1350** and explain how TFTP recovers from transmission errors and losses.
11. Is it possible for a go-back-n receiver to interoperate with a selective-repeat sender ? Justify your answer.
12. Is it possible for a selective-repeat receiver to interoperate with a go-back-n sender ? Justify your answer.
13. The go-back-n and selective repeat mechanisms that are described in the book exclusively rely on cumulative acknowledgements. This implies that a receiver always returns to the sender information about the last segment that was received in-sequence. If there are frequent losses or reordering, a selective repeat receiver could return several times the same cumulative acknowledgment. Can you think of other types of acknowledgements that could be used by a selective repeat receiver to provide additional information about the out-of-sequence segments that it has received. Design such acknowledgements and explain how the sender should react upon reception of this information.
14. The *goodput* achieved by a transport protocol is usually defined as the number of application layer bytes that are exchanged per unit of time. What are the factors that can influence the *goodput* achieved by a given transport protocol ?
15. The Transmission Control Protocol (TCP) attaches a 40 bytes header to each segment sent. Assuming an infinite window and no losses nor transmission errors, derive a formula that computes the maximum TCP goodput in function of the size of the segments that are sent.
16. A go-back-n sender uses a window size encoded in a *n* bits field. How many segments can it send without receiving an acknowledgement ?
17. Consider the following situation. A go-back-n receiver has sent a full window of data segments. All the segments have been received correctly and in-order by the receiver, but all the returned acknowledgements have been lost. Show by using a time sequence diagram (e.g. by considering a window of four segments) what happens in this case. Can you fix the problem on the go-back-n sender ?
18. Same question as above, but assume now that both the sender and the receiver implement selective repeat. Note the the answer will be different from the above question.
19. Consider a transport that supports window of one hundred 1250 Bytes segments. What is the maximum bandwidth that this protocol can achieve if the round-trip-time is set to one second ? What happens if, instead of advertising a window of one hundred segments, the receiver decides to advertise a window of 10 segments ?
20. Explain under which circumstances a transport entity could advertise a window of 0 segments ?
21. The socket library is also used to develop applications above the reliable bytestream service provided by TCP. We have installed on the *sirius.info.ucl.ac.be* server a simple server that provides a simple client-server service. The service operates as follows :
    - the server listens on port *62141* for a TCP connection
    - upon the establishment of a TCP connection, the server sends an integer by using the following TLV format :
        - the first two bits indicate the type of information (01 for ASCII, 10 for boolean)
        - the next six bits indicate the length of the information (in bytes)

  - An ASCII TLV has a variable length and the next bytes contain one ASCII charater per byte. A boolean TLV has a length of one byte. The byte is set to *00000000b* for *true* and *00000001b* for false.

- the client replies by sending the received integer encoded as a 32 bits integer in *network byte order*
- the server returns a TLV containing *true* if the integer was correct and a TLV containing *false* otherwise and closes the TCP connection

Each group of two students must implement a client to interact with this server in C, Java or python.

CHAPTER

THREE

PROGRAMMING PROJECT

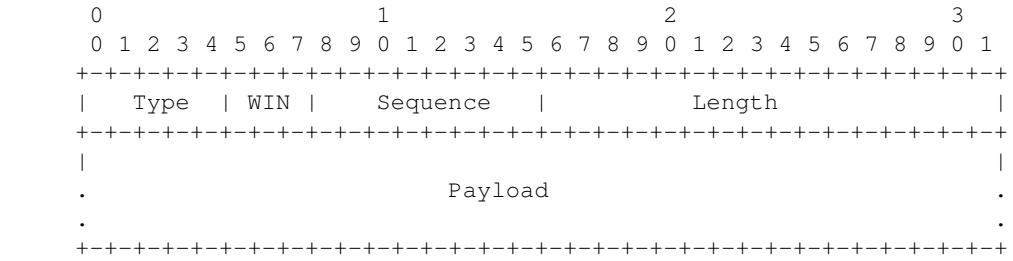
Your objective in this project is to implement a simple reliable transport protocol by groups of 2 students. These groups must be subgroups of the main groups that are registered on icampus. If the number of students in an icampus group is odd, there can be one group of three students.

The protocol uses a sliding window to transmit more than one segment without being forced to wait for an acknowledgment. Your implementation must support variable size sliding window as the other end of the flow can send its maximum window size. The window size is encoded as a three bits unsigned integer.

The protocol identifies the DATA segments by using sequence numbers. The sequence number of the first segment must be 0. It is incremented by one for each new segment. The receiver must acknowledge the delivered segments by sending an ACK segment. The sequence number field in the ACK segment always contains the sequence number of the next expected in-sequence segment at the receiver. The flow of data is unidirectional, meaning that the sender only sends DATA segments and the receiver only sends ACK segments.

To deal with segments losses, the protocol must implement a recovery technique such as go-back-n or selective repeat and use retransmission timers. The project will partially be evaluated on the quality of the recovery technique. Groups of three must implement the selective repeat technique while groups of two can implement a simpler recovery scheme such as go-back-n.

```
Segment format
```



- *Type*: segment type
  - 0x1 DATA segment.
  - 0x2 ACK segment
- *WIN*: the size of the current window (an integer encoded as a 3 bits field). In DATA segments, this field indicates the size of the sending window of the sender. In ACK segments, this field indicates the current value of the receiving window.
- *Sequence*: Sequence number (8 bits unsigned integer), starts at 0. The sequence number is incremented by 1 for each new DATA segment sent by the sender. Inside an ACK segment, the sequence field carries the sequence number of the next in-sequence segment that is expected by the receiver.
- *Length*: length of the payload in multiple of one byte. All DATA segments contain a payload with 512 bytes of data, except the last DATA segment of a transfert that can be shorter. The reception of a DATA segment whose length is different than 512 indicates the end of the data transert.
- Payload: the data to send

3.1 Deliverables

Before October 22nd, 2010 at 23:59 each sub-group must submit its commented source code (with a Makefile) on the SVN and a short report (up to four pages in pdf format) describing the chosen recovery technique, the architecture of the client and server and the tests that have been carried out. Each group must implement both a receiver and a sender. The implementation language can be chosen among C, Java and Python.

The client and the server exchange UDP datagrams that contain the DATA and ACK segments. They must be command-line tools that allow to transmit one binary file and support the following parameters :

```
sender <destination_DNS_name> <destination_port_number> <window_size> <input_file>

receiver <listening_port_number> <window_size> <output_file>
```

A demo session will be organised on Tuesday October 26th. During the demo session, you will be invited to demonstrate that your implementation is operational and is interoperable with another. You also need to perform tests to show that your implementations works well in case of segment losses. For these tests, you can use a random number generator to probabilistically drop received segments and introduce random delays upon the arrival of a segment.

CHAPTER

FOUR

INDICES AND TABLES

- *genindex*
- *search*

INDEX

R