Networking scenarios

DaCoPAn

Helsinki 11th April 2005

Software Engineering Project

UNIVERSITY OF HELSINKI	UNIVERSITY OF PETROZAVODSK
Department of Computer Science	Department of Computer Science

Course
581260 Software Engineering Project (6 cr)

Project Group
Carlos Arrastia Aparicio
Jari Aarniala
Alejandro Fernandez Rey
Vesa Vainio
Jarkko Laine
Jonathan Brown

Kirill Kulakov
Andrey Salo
Andrey Ananin
Mikhail Kryshen
Viktor Surikov

Customer
Markku Kojo

Project Masters
Juha Taina (Supervisor)
Yury Bogoyavlenskiy (Supervisor)

Turjo Tuohiniemi (Instructor)
Dmitry Korzun (Instructor)

Homepage
`http://www.cs.helsinki.fi/group/dacopan`

Change Log

Version	Date	Modifications
1.0	Put the date here	First version

Contents

1 Networking scenario format

This section describes the meanings of the fields in the networking scenario format that is used to document all the networking scenarios for the project. Note that some fields are reserved for particular phases of the project, and all fields need not be filled when the scenarios are initially documented.

The most important fields of each scenario are the top-level Description of mechanism field and the Variables field. These define the technical content that should be understood by the developers. They also define information that student user should learn from the scenario and thus define the educational objective by default (the educational objective can be qualified or constrained by comments in the Educational objective field).

- Number: Number of the scenario. Each number should be unique, and persistent all through the project's life span. The numbers informally reflect a hierarchical organization, where the hundreds digit shows the highest relevant layer for the scenario and the tens digit shows e.g. the particular protocol that is used in scenario (or feature of protocol).
- Name: Unique name for the scenario. May be changed if needed.
- Implementation priority: If and when decisions are made about implementation priority for the scenario, this field may be used to hold this information. Should not be filled before formal decision is made.
- Description of mechanism: Describes "What goes on?" in the scenario. A detailed description of the networking mechanisms and events that this scenario is meant to describe. Description should be as detailed as possible, including also any mechanisms outside the main interest area of the mechanism. An important objective is that when someone understands the mechanism involved in the scenario, he/she can give a detailed explanation for everything that happens in the scenario.
- Relevant layers: The layers that are relevant to the mechanism should be listed here. Allowed items in the list are: Application layer / Transport layer / Network layer / Link layer.
- Educational objective: By default, the educational objective is that the student user understands the mechanism described on Description of mechanism. If there is need to make comments in addition to this default assumption, they should be written in this field. Otherwise this field may be left blank. It is not desirable to fill in self-evident or redundant information here.
- Application layer / Transport layer / Network layer / Link layer: These are not fields, they are headers for layer-specific sections. Fields in each layer-specific section are: Description of mechanism and Information to visualize.

  - Description of mechanism: A description of what happens on the scenario on the particular layer. Probably should be a subset of the top-level description of mechanism.
  - Protocol header fields: Description of any header field values or end point internal variables that contribute to the mechanism of this scenario. The list should be complete, including information about values or variables that contribute to the activity seen in tcpdump logs in any way. The names of variables should be consistent across all scenarios.
  - Variables: Information that affects how the hosts work, but is not explicitly expressed in the protocol headers.
  - Information to visualize: Description of information that the animator program should be able to present visually.

- Program configuration: To be filled during the implementation phase of the project. Description of configurations or settings of the animator that should be used to animate this scenario.

- Instructions for user: Optional field. In case it is desirable to record what instructions need to be given for the student end-user of the animator program, this information can be recorded here. Probably this is relevant when the educational use of each scenario is designed in detail.

- Pair of tcpdump files: When the Customer provides the project group with tcpdump files for the scenarios. The filenames of these files should be filled in the scenario descriptions. Project group is allowed to rename files and develop a consistent naming scheme for the files. After the filenames have been filled in the scenario descriptions, these names should be used consistently all through the project.

2 Networking scenarios

2.1 Notes

- All the following networking scenarios need as variables timestamps, so it will not be present in the following description, but this variable is needed.

- The names for protocol header fields are the used in their respective RFCs

- The protocol header fields in the layers description contain the header fields which are relevant for educational purposes, but the protocol events file should contain, if possible, all the header fields of the protocol.

- By default the following scenarios try to be as simplified as possible and do not support delayed ACKs nor DNS lookups nor connection establishment. Unless that it is explicitly written.

- Some of the scenarios presented with priority 3 do not present list of variables nor protocol header fields. They should be filled in revisions of the document for future improvements.

2.2 Implementation priority #1

2.2.1 201 - IP packet delivery and content

Description: A sends an IP datagram to B.
Relevant layers: Network layer
Educational objectives: Show how data is transferred in network layer and learn about the fields used in IP headers to transfer the datagram from one endpoint to the other.

Network layer.

Description of mechanism	A wants to send a piece of information to B. The size of this data is less than the MTU, and fits in an IP datagram. The header of the IP datagram is filled with the corresponding values (Total Length, Identification, Flags, Offset, IP Addresses, Version, Header Length, TTL, Checksum and options) and sent to B.
Protocol header fields	Total Length, Datagram Identification, Flags (DF=1, MF), Offset, Source IP Address, Destination IP Address, *Version*, *Header Length*, *Time To Live*, *Checksum*
Variables	MTU
Information to visualize	Fields present in the IP header and transfer of information from endpoint A to B

Additional information.

Program configuration	
Instructions for the user	
tcpdump filenames	

2.2.2 310 - IP fragmentation and simple UDP data transfer

Description: This scenario combines IP fragmentation and simple UDP transfer. A sends an UDP packet to B. This is sent using IP and the size of the data to be sent exceeds the MTU value of the network. The data is fragmented into some IP datagrams and sent to B. B receives them and proceeds to their defragmentation. At this point the original UDP packet sent is received in B.
Relevant layers: Network layer, Transport layer
Educational objectives: Show how the different layers work together to transfer data to the other endpoint to understand this might be included the concepts of destination port, present in the UDP header. Visualize how UDP works, a transport protocol without ack-owledgements. Show how IP datagrams are fragmented in the sending endpoint. Show also how these datagrams are defragmented in the receiving endpoint.

Transport layer.

Description of mechanism	An UDP datagram is sent from A to B. The destination port and size of the packet is filled in this layer.
Protocol header fields	Source Port, Destination Port, Length
Variables	-
Information to visualize	How packets are transferred from endpoint A to B. Header fields and notices of fragmentation in the network layer.

Network layer.

Description of mechanism	A wants to send a piece of information to B. The size of this data is bigger than the MTU, and doesn't fit in an IP datagram. The header fields of the IP datagram are filled with the corresponding values and sent to B. From one datagram to another the values that change in the header fields are the Flags MF (More Fragments set to 1 if the fragment is the last one), DF (0 when there is fragmentation) and Offset of the data in the original datagram
Protocol header fields	Total Length, Datagram Identificator, Flags (MF, DF), Offset
Variables	MTU value
Information to visualize	Process of fragmentation and defragmentation. Header fields that change from one datagram to other and also header fields not differring between datagrams.

Additional information.

Program configuration	
Instructions for the user	
tcpdump filenames	

2.2.3 321 - Three-way TCP connection establishment

Description: Endpoint A (client) wants to start a communication with B (server) to transfer information using the TCP protocol. Then uses the three-way TCP connection establishment mechanism to assure that the server can attend its request. Three TCP packets are sent to the network in this process. The first one from A to B specifying that A wants to connect to B and the port to do it. In the second the availability of B is confirmed to A by an acknowledgement. Finally A acknowledges B the reception of its last packet sent. At this point the TCP connection is established. Maximum segment size negotiation between both hosts should be visualized.
Relevant layers: Transport layer
Educational objectives: The student should learn the mechanism of TCP connection establishment and how all the fields and variables involved in this process change.

Transport layer.

Description of mechanism	The requesting end (client) sends a SYN segment specifying the port number of the server that the client wants to connect to, and the client's initial sequence number (ISN). The server responds with its own SYN segment containing the server's initial sequence number. The server also acknowledges the client's SYN by ACKing the client's ISN plus one. A SYN consumes one sequence number. The client must acknowledge this SYN from the server by ACKing the server's ISN plus one. In the first two steps of this processthe endpoints communicate each other their respective Maximum Segment Size MSS.
Protocol header fields	Sequence Number, Acknowledgment Number, Flags (SYN, ACK), Maximum Segment Size (Options with Kind=3)
Variables	TCP states
Information to visualize	SYN and ACK flags and numbers in packets moving between the two endpoints. Also the different host states the two enpoints have after sending or receiving the packets. Maximum segment size negotiation based in the maximum segment size sent by the 'server' endpoint.

Additional information.

Program configuration	
Instructions for the user	
tcpdump filenames	

2.2.4 341 - TCP normal FIN termination

Description: A client host wants to end the TCP connection existing between it and a server host. Four packets are needed to close the connection. Client sends FIN packet to server (sequence number x), which acknowledges (sequence number x+1). Then server sends FIN packet to client (sequence number y), then client acknowledges (sequence number y+1). The connection is then closed.
Relevant layers: Transport layer
Educational objectives: The student should learn which is the packet interchange correspondant to this half-close mechanism.

Transport layer.

Description of mechanism	Client sends FIN packet to server (sequence number x), which acknowledges (sequence number x+1). Then server sends FIN packet to client (sequence number y), then client acknowledges (sequence number y+1). The connection is then closed.
Protocol header fields	Sequence Number, Acknowledgment Number, Flags (FIN, ACK)
Variables	TCP states
Information to visualize	Key header fields for each packet. TCP states of both hosts during the process of closing the connection.

Additional information.

Program configuration	
Instructions for the user	
tcpdump filenames	

2.2.5 361 - Slow start and sending data with TCP

Description: A TCP connection has been established between 2 hosts. The slow start algorithm controls TCP flow. It operates by observing that the rate at which new packets should be injected into the network is the rate at which the acknowledgments are returned by the other end, using the congestion window, called cwnd.
Relevant layers: Transport layer
Educational objectives: The student should learn how the slow start algorithm works: how cwnd is incremented by each arrival of an ACK packet, and how several packets can be sent in a row and acknowledged by just one ACK, depending on the value of cwnd.

Transport layer.

Description of mechanism	One host sends data to the other, which acknowledges it. For every acknowledgment, the congestion window value is increased by 1/2/... Maximum Segment Size's (MSS). This means that if the cwnd is 3*MSS, the sender can send 3 packets with MSS. The congestion (when the receiver can't process that many packets sent in a row) is not visualized in this scenario.
Protocol header fields	Sequence Number, Acknowledgment Number, Flags (ACK), Maximum Segment Size (Options with Kind=3)
Variables	Congestion window (cwnd)
Information to visualize	Key header fields for each packet. If possible, the value of cwnd can be shown.

Additional information.

Program configuration	
Instructions for the user	
tcpdump filenames	

2.2.6 362 - TCP Recovery through fast retransmit (happening in slow start with packet loss)

Description: This scenario starts with the normal slow start defined in the scenario number 361 and once a packet is lost the TCP fast retransmit algorithm starts. TCP is required to generate an immediate acknowledgment when an out-of-order segment is received. This duplicate ACK should not be delayed. The purpose of this duplicate ACK is to let the other end know that a segment was received out of order, and to tell it what sequence number is expected. From A point of view, duplicate ACK is caused by a lost segment or just a reordering of segments, so it waits for a small number of duplicate ACKs (dupack threshold) to be received. If three or more duplicate ACKs are received in a row, it is a strong indication that a segment has been lost. The client then performs a retransmission of what appears to be the missing segment, without waiting for a retransmission timer to expire. Slow start is not performed after the three duplicate ACKs are received because the receipt of the duplicate ACKs tells the client that a packet has been lost. Since the receiver can only generate the duplicate ACK when another segment is received, that segment has left the network and is in the receiver's buffer. That is, there is still data flowing between the two ends, and we don't want to reduce the flow abruptly by going into slow start. Next, congestion avoidance, but not slow start is performed.
Relevant layers: Transport layer
Educational objectives: The student should learn what happens if a packet is lost during the slow start and also how TCP can re-transmit lost packets ina an efficient way without waiting for any retransmission time to occur.

Transport layer.

Description of mechanism	The client starts the normal slow start as defined in the scenario 361. When the receiver (B) gets a packet with unexpected sequence number it knows that a packet has been lost.?A waits for a small number of duplicated ACKs (dupack threshold. Usually 2)A total of 3 duplicate ACKs are sent for the lost packet (4 ACKs in total). When A receives the 3rd duplicate ACK, it knows that B never received the packet B's been sending the duplicate ACKs for, so A re-sends the lost packet. In this process the following variables are modified in the following way: 1. When the third duplicate ACK is received, set ssthresh to one-half the current congestion window, cwnd. Retransmit the missing segment. Set cwnd to ssthresh plus 3 times the segment size. 2. Each time another duplicate ACK arrives, increment cwnd by the segment size and transmit a packet (if allowed by the new value of cwnd). 3. When the next ACK arrives that acknowledges new data, set cwnd to ssthresh (the value set in step 1). This should be the ACK of the retransmission from step 1, one round-trip time after the retransmission. Additionally, this ACK should acknowledge all the intermediate segments sent between the lost packet and the receipt of the first duplicate ACK. This step is congestion avoidance, since we're slowing down to one-half the rate we were at when the packet was lost.
Protocol header fields	Sequence Number, Acknowledgment Number, Flags (ACK), Maximum Segment Size (Options with Kind=3)
Variables	Congestion Window (cwnd), slow start threshold size (ssthresh), Retransmission TimeOut (RTO), Duplicated ACK (dupack) Threshold
Information to visualize	The lost packets, the duplicate ACKs, the values of ssthresh and cwnd and how they change. The way the client react to the lost packets.

Additional information.

Program configuration	
Instructions for the user	
tcpdump filenames	

2.3 Implementation priority #2

2.3.1 333 - TCP packet loss and recovery through RTO

Description: The idea is to show a simple timeout and retransmission example. A (the client) sends a TCP packet in an established TCP connection to B (server). If after a given timeot expires (RTO: Retransmission Timeout) the acknowledgement for the packet has not been received, the packet is sent again. In the meanwhile B is waiting for more packets to arrive, if a new packet is received it answers with an acknowledgement packet.
Relevant layers: Transport layer
Educational objectives: Show the mechanism of TCP to overcome packet loss with retransmission. Introduce RTO concept and its use

Transport layer.

Description of mechanism	Client A is sending TCP packets to server B. Then after sending a packet A waits for the ACK of that packet to arrive, but this doesn't happen. After RTO (Retransmission Timeout) has expired, A retransmit the packet. Two are the reasons because the ACK doesn't arrive to A: Packet loss or ACK loss. In both cases TCP can handle the situation. Here is important to choose a suitable value of RTO. Small values of RTO could cause unnecessary retransmissions. RTO value depend of another time value called RTT (Round Trip Time) measuring the time passed between the moment when a packet is sent and its ACK is received. The RTO value is increased in successive attempts of retransmission for the same packet. This increment is exponential. The connection is closed in case that more packets are lost and B doesn't receive them.
Protocol header fields	Sequence Number, Acknowledgment Number, Flags (ACK)
Variables	Retransmission TimeOut (RTO)
Information to visualize	Packet loss and/or acknowledgement lost, RTO value and how it change in successive loss of the same packet. Transfer continuation after successful retransmission. Connection closed in case that more packets are lost and B doesn't receive them before the keepalive timeout

Additional information.

Program configuration	
Instructions for the user	
tcpdump filenames	

2.3.2 411 - HTTP: Simple request with reply

Description: A client (web browser) requests an HTML page from a server and receives it using HTTP protocol. No embedded objects will be included in the transfered HTML page (such as images, ...).
Relevant layers: Application layer
Educational objectives: Show the basic behavior of the HTTP protocol (request and reply), and how this protocol uses TCP on the transport layer.

Application layer.

Description of mechanism	Host A (client) sends an HTTP request message to host B (server). Host B then answers with an HTTP containing an HTML large enough to require a few TCP segments.
Protocol header fields	**Request** = Request line + general header + request header + entity header + message body **Response** = Status line + general header + entity header + message body
Variables	-
Information to visualize	HTTP headers both in the request and reply messages.

Transport layer.

Description of mechanism	The request message from A is put in a TCP packet and sent to B using the mechanisms defined in other scenarios. The reply from B is divided into several TCP packets that are sent to A. For more information see scenario number 331.
Protocol header fields	-
Variables	-
Information to visualize	The encapsulation of HTTP messages to TCP packets.

Additional information.

Program configuration	
Instructions for the user	
tcpdump filenames	

2.3.3 101 - ARP Lookup

Description: A has the IP address of B and wants to know B's physical ethernet address. A sends broadcast ARP request for the address of B, B recognizes its IP address and answers to A.
Relevant layers: ARP layer
Educational objectives: Show when machines use the mechanism for address resolution within a network

ARP layer.

Description of mechanism	See general description above
Protocol header fields	Type of message (request/response), physical and network addresses of source and destination
Variables	-
Information to visualize	Notification about the presence of this request

Additional information.

Program configuration	
Instructions for the user	
tcpdump filenames	

2.3.4 322 - TCP Timeout of Connection Establishment

Description: During an attempt of connection establishment it can happen that the server doesn't answer after beginning a connection establishment request. The client has to handle this sutuation and set a timeout to exit if the reply to its request is not received.
Relevant layers: Transport layer
Educational objectives: Show how TCP reacts when a connection establishment is initiated and no replay is received (from the point of view of the client)

Transport layer.

Description of mechanism	A sends SYN to B and goes to state SYN SENT. If B doesn't get the SYN, it stays in the LISTEN state, and A will go to the state CLOSED after a timeout expires. If B gets the SYN and sends an ACK, it goes to state SYN RCVD. If A doesn't get the ACK, its timeout will expire and it'll go to state CLOSED. If A gets the ACK, it goes to ESTABLISHED and sends ACK to B. If B doesn't get A's ACK, its timeout expires and it goes to LISTEN.
Protocol header fields	
Variables	TCP states
Information to visualize	States of the hosts and different packets sent (and lost in case it happens). Timeout to go to LISTEN state by the client

Additional information.

Program configuration	
Instructions for the user	
tcpdump filenames	

2.4 Implementation priority #3 (Postponed networking scenarios)

2.4.1 351 - TCP Connection reset (RST)

Description: In general, a reset is sent by TCP whenever a segment arrives that doesn't appear correct for the referenced connection. This can happen in the following three situations:
* Connection Request to Nonexistent Port
* Aborting a connection
* Detecting Half Open Connections
Relevant layers: Transport layer
Educational objectives: Show the conditions in which TCP reacts to reset the connection and the way to do it

Transport layer.

Description of mechanism	The reset segments are sent in these three situations: * Connection Request to Nonexistent Port: a connection request arrives and no process is listening on the destination port. The server sends to the client a reset segment to inform about this fact. * Aborting a connection: It is possible to abort a connection by sending a reset instead of a FIN. This is sometimes called an abortive release. TCP on sends an RST instead of the normal FIN. The RST segment contains a sequence number and acknowledgment number. Also notice that the RST segment elicits no response from the other endit is not acknowledged at all. The receiver of the reset aborts the connection and advises the application that the connection was reset. * Detecting Half Open Connections: A RST segment is sent by one of the ends to determine if the other end is still connected
Protocol header fields	
Variables	TCP states
Information to visualize	RST segments and situations in which are sent

Additional information.

Program configuration	
Instructions for the user	
tcpdump filenames	

2.4.2 412 - HTTP: Retrieval with multiple connections

Description: An HTML page with a couple of images is downloaded from the server. The data is downloaded using several connections, one for each file.
Relevant layers: Application layer, Transport layer
Educational objectives: The student should see how embedded objects (inline images) are fetched using new TCP connections.

Application layer.

Description of mechanism	The procedure is similar to that of scenario 411, but instead of sending just one query to the server, the clients sends more. It first sends a request message asking for the HTML page. The server replies by sending a reply with that page in it. When the client (browser) notices that there are image tags in the file, it sends new requests to the server, asking for those images to be sent.
Protocol header fields	**Request** = Request line + general header + request header + entity header + message body **Response** = Status line + general header + entity header + message body
Variables	-
Information to visualize	The different requests and replies between client and server. The inline image tags in the HTTP message contents (HTML page) could be also presented somehow.

Transport layer.

Description of mechanism	On the TCP level downloading these new files result as new connections between the client and server. For each file a connection is created (see scenario 321) and then the packets are sent between these end points.
Protocol header fields	-
Variables	-
Information to visualize	Different connections should be somehow presented separately from each other, for example by using color.

Additional information.

Program configuration	
Instructions for the user	
tcpdump filenames	

2.4.3 421 - DNS request for A record

Description: Some application wants to find out the IP address for some domain name. It sends the address to a DNS server that then replies with the correct address record.
Relevant layers: Application layer
Educational objectives: The student should learn how an application can query a DNS server and receive a response.

Application layer.

Description of mechanism	Host A sends a DNS query containing the domain name to be resolved to a 3rd party acting as a name server. The name server then responds with address information (A record). In case the name server doesn't contain the A record for this domain name, it contacts another name server to ask for it. And so on.
Protocol header fields	Identification, flags, number of questions, number of answer RRs, number of authority RRs, number of additional RRs.
Variables	Header fields: questions and answers
Information to visualize	The requested domain name in request and address information in reply.

Additional information.

Program configuration	
Instructions for the user	
tcpdump filenames	