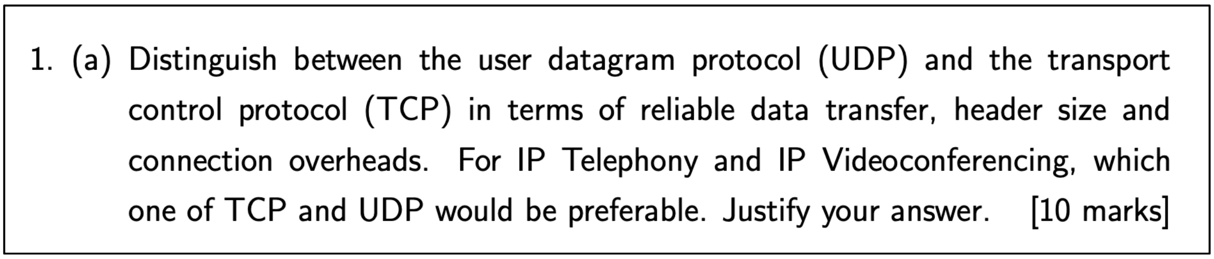
CS3031 – Advanced Telecommunications
Sample Exam Solutions

Question 1

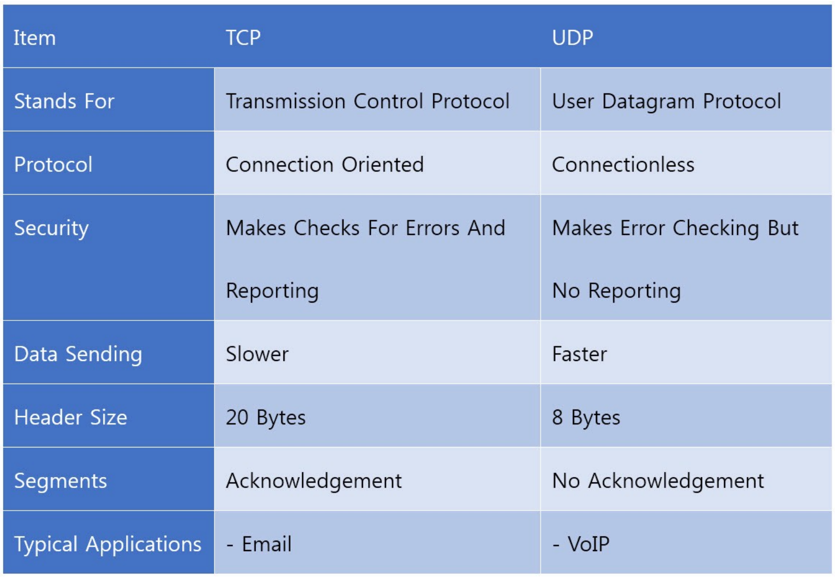


1. (a) Distinguish between the user datagram protocol (UDP) and the transport control protocol (TCP) in terms of reliable data transfer, header size and connection overheads. For IP Telephony and IP Videoconferencing, which one of TCP and UDP would be preferable. Justify your answer. [10 marks]

User Datagram Protocol (UDP): This is a 'bare bones' internet transport protocol. It is a *'connectionless'* protocol which means there is no need to establish a prior connection between the two communicating nodes. It has a very small header size as a result. There is no congestion control within UDP meaning it can blast away as fast as it likes.

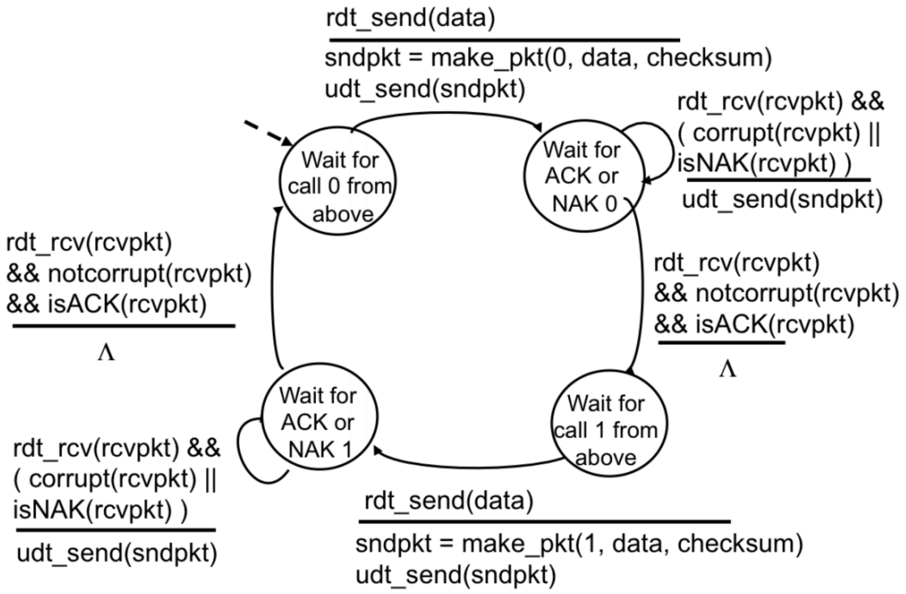
Transport Control Protocol (TCP): This is a connection-orientated protocol. It is introduces a reliable, in-order byte stream of communication using sequence numbers. It introduces a congestion control mechanism. When two nodes would like to communicate they must undergo a three-way TCP handshake in order to initialize the communication. TCP headers are much larger than UDP as they need to carry additional information such as SEQ and ACK numbers, checksums and receive window lengths. All of these additional parameters allow TCP to serve as a much more reliable form of transport.

For IP Telephony and IP Videoconferencing you would more than likely prefer to use UDP. This is because these forms of communication do not require 100% accurate packet transmission and can afford for lossy data to occur. If we were to use TCP for these communication methods there would be an additional latency in establishing connection and recovering lost packets. By using TCP your calls would constantly be lagging/stalled trying to recover the packets that were lost in transit.



Item	TCP	UDP
Stands For	Transmission Control Protocol	User Datagram Protocol
Protocol	Connection Oriented	Connectionless
Security	Makes Checks For Errors And Reporting	Makes Error Checking But No Reporting
Data Sending	Slower	Faster
Header Size	20 Bytes	8 Bytes
Segments	Acknowledgement	No Acknowledgement
Typical Applications	- Email	- VoIP

(b) The diagram below shows the finite state machine (FSM) for a reliable sender that can handle garbled ACKs and NAKs. Draw the FSM for the corresponding receiver.


rdt_send(data)
sndpkt = make_pkt(0, data, checksum)
udt_send(sndpkt)
Wait for call 0 from above
Wait for ACK or NAK 0
rdt_rcv(rcvpkt) && (corrupt(rcvpkt) || isNAK(rcvpkt))
udt_send(sndpkt)
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isACK(rcvpkt)
Λ
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && isACK(rcvpkt)
Λ
Wait for ACK or NAK 1
Wait for call 1 from above
rdt_rcv(rcvpkt) && (corrupt(rcvpkt) || isNAK(rcvpkt))
udt_send(sndpkt)
rdt_send(data)
sndpkt = make_pkt(1, data, checksum)
udt_send(sndpkt)

[12 marks]

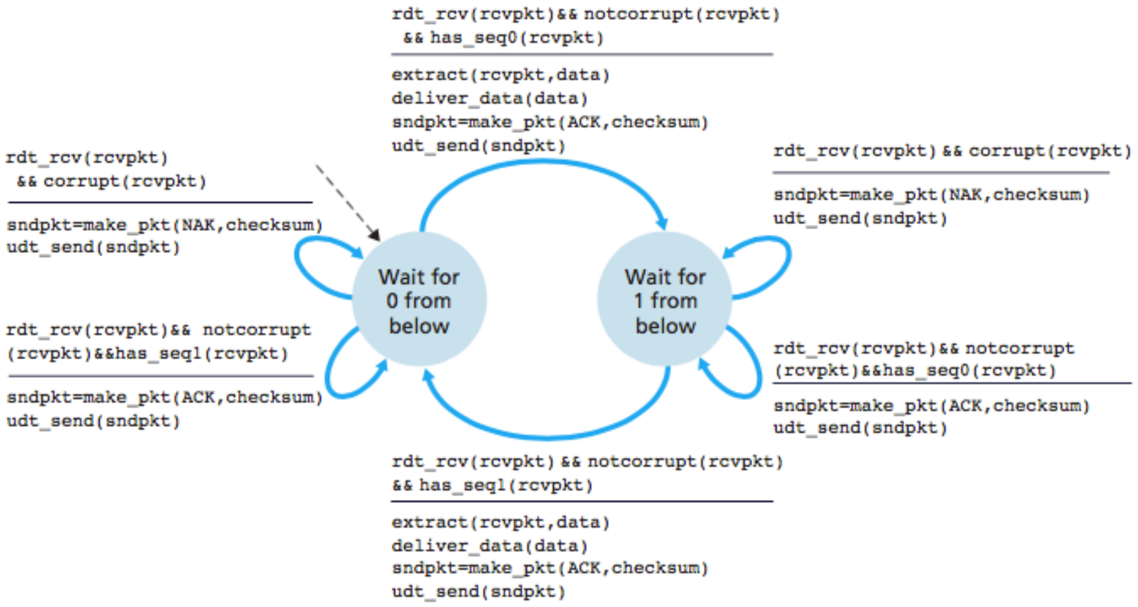
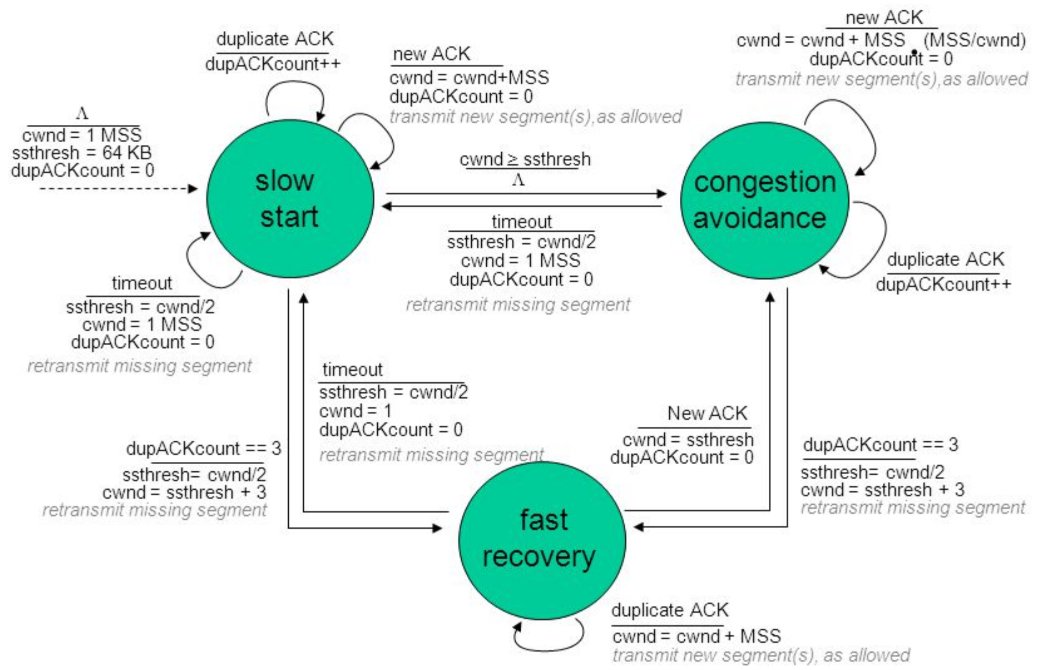

rdt_rcv(rcvpkt)&& notcorrupt(rcvpkt) && has_seq0(rcvpkt)
extract(rcvpkt,data)
deliver_data(data)
sndpkt=make_pkt(ACK,checksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) && corrupt(rcvpkt)
sndpkt=make_pkt(NAK,checksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) && corrupt(rcvpkt)
sndpkt=make_pkt(NAK,checksum)
udt_send(sndpkt)
Wait for 0 from below
Wait for 1 from below
rdt_rcv(rcvpkt)&& notcorrupt (rcvpkt)&&has_seq1(rcvpkt)
sndpkt=make_pkt(ACK,checksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt)&& notcorrupt (rcvpkt)&&has_seq0(rcvpkt)
sndpkt=make_pkt(ACK,checksum)
udt_send(sndpkt)
rdt_rcv(rcvpkt) && notcorrupt(rcvpkt) && has_seq1(rcvpkt)
extract(rcvpkt,data)
deliver_data(data)
sndpkt=make_pkt(ACK,checksum)
udt_send(sndpkt)

Figure 3.12 ♦ rdt2.1 receiver

(c) With the help of a FSM describe the main building blocks of the TCP Congestion Control algorithm. You are required to detail the state transitions in the FSM. [12 marks]

TCP congestion control FSM: details



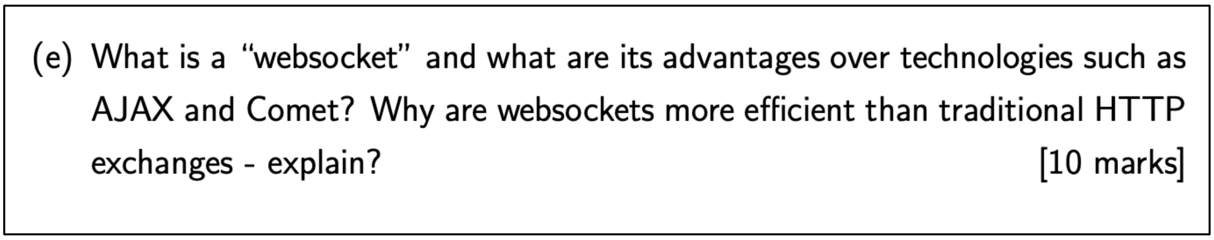
Transport Layer 3-4

(d) What is the role of a "web proxy" in a large institutional network? What are the main advantages of installing a proxy server? [6 marks]

The role of a web proxy in a large institutional network is to serve as an intermediary between incoming connections and the destination servers.

Proxy servers help improve web performance by storing a copy of frequently used webpages in a local cache on the proxy server. When a client requests a webpage stored in the proxy server's cache, it is provided by the proxy server which is faster than relaying the request to the desired web server to fetch the page on every occurrence. Proxy servers also help improve security by filtering out some web content and malicious software.

Proxy servers also allow for requests to be blocked dynamically as well as sites, protocols and ports.



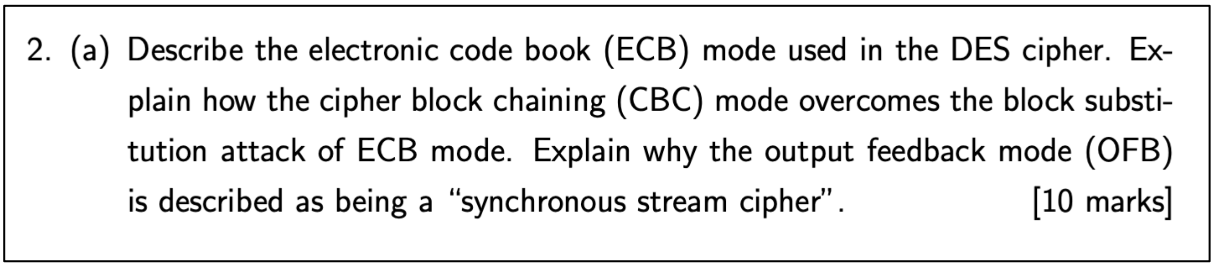
(e) What is a “websocket” and what are its advantages over technologies such as AJAX and Comet? Why are websockets more efficient than traditional HTTP exchanges - explain? [10 marks]

Websockets allow a long-held TCP socket connection to be established between the client and server which allows bi-directional, full duplex, messages to be instantly distributed with little overhead resulting in a very low latency connection.

Websockets don’t fully replace AJAX, which should still be used for making short-lived web service calls. Unlike AJAX, WebSockets are based around an event model. The client and server can emit events and send data to each other whenever they want to. Whilst with AJAX each request must be followed by a response in a one-directional manner. Similar to AJAX, WebSockets must establish a connection with the server, however WebSockets only have to perform this once whilst with AJAX it must be done with every request causing high overhead latency.

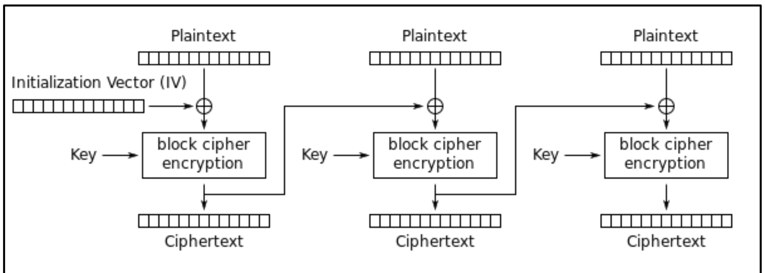
Comet is a protocol that allows ‘**Sever push**’ behaviour wherein a server pushes events to a clients browser. It allows for the server to stream events to a client over a single persistent connection. It also allows for long polling where the browser polls the server for new events with a persistent request that is held open until it gets a response. WebSockets can be considered an extension upon Comet that allows for bi-directional pushing of events both to and from the client and server. This removes the need for long polling and significantly reduces the combined overhead of the communication.

Question 2


2. (a) Describe the electronic code book (ECB) mode used in the DES cipher. Explain how the cipher block chaining (CBC) mode overcomes the block substitution attack of ECB mode. Explain why the output feedback mode (OFB) is described as being a "synchronous stream cipher". [10 marks]

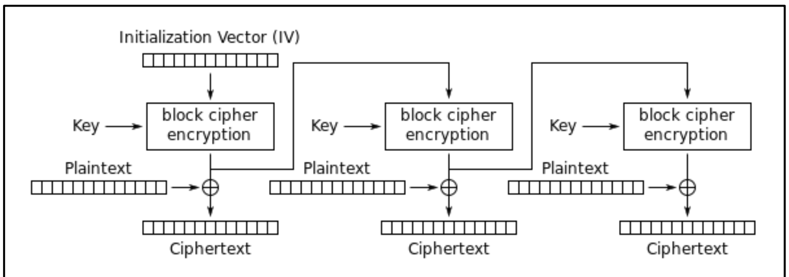
Electronic Code Book (ECB) mode used in the DES cipher is when the plaintext is operated upon independently in 64 bit (8 byte) chunks at a time. Using a 56-bit key and the DES encryption algorithm each 64 bit chunk is encrypted. This can be performed in parallel. However, this is a bad thing as ECB mode encrypts in a highly deterministic manner. Identical plain text blocks result in identical ciphertext blocks. This can allow for a substitution attack to occur as each block is independent of any other block. Therefore, no one will notice if an attacker changes any of the blocks (i.e bank address).

Cipher Block Chaining (CBC) mode is when each block of plaintext is XOR'd with the previous ciphertext block after it has been encrypted. The initial block is XOR'd with an Initialisation Vector (IV). This mode introduces a dependency as each ciphertext block depends on all plaintext blocks processed up to that point. If one block is changed, the resulting following ciphertext blocks will be corrupted.

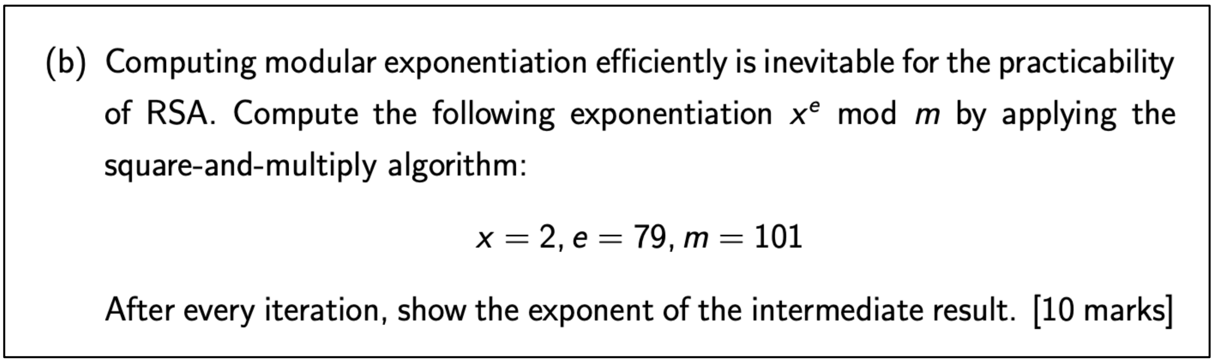


Cipher Block Chaining (CBC) mode encryption

Output Feedback Mode (OFB) is described as being a *"synchronous stream cipher"* as within OFB a block cipher encryption is used to create a stream cipher encryption scheme. The output of the cipher outputs b key stream bits, where b is the width of the block cipher used. Each plaintext digit is then encrypted one at a time with the corresponding digit in the keystream.



Output Feedback (OFB) mode encryption



(b) Computing modular exponentiation efficiently is inevitable for the practicability of RSA. Compute the following exponentiation x^e mod m by applying the square-and-multiply algorithm:

$$x = 2, e = 79, m = 101$$

After every iteration, show the exponent of the intermediate result. [10 marks]

The aim of the game here is to compute:

$$2^{79}\ mod\ 101$$

First compute 2^1:

$$2^1 = 2 \rightarrow 2\ mod\ 101$$

Now square both sides of the exponentiation:

$$2^2 = (2^1)^2$$

$$2^2 = 4 \rightarrow 4\ mod\ 101$$

Square both sides again:

$$2^4 = (2^2)^2$$

$$2^4 = 16 \rightarrow 16\ mod\ 101$$

Square both sides again:

$$2^8 = (2^4)^2$$

$$2^8 = 256 \rightarrow \boldsymbol{54\ mod\ 101}$$

Square both sides again:

$$2^{16} = (2^8)^2$$

$$2^{16} = (\mathbf{54})^{\mathbf{2}}\ \boldsymbol{mod\ 101}$$

$$2^{16} = \boldsymbol{2916\ mod\ 101}$$

$$2^{16} = \boldsymbol{88\ mod\ 101}$$

Square both sides again:

$$2^{32} = (2^{16})^2$$

$$2^{32} = (\mathbf{88})^{\mathbf{2}}\ \boldsymbol{mod}\ \mathbf{101}$$

$$2^{32} = \mathbf{7744}\ \boldsymbol{mod}\ \mathbf{101}$$

$$2^{32} = \mathbf{68}\ \boldsymbol{mod}\ \mathbf{101}$$

Square both sides again:

$$2^{64} = (2^{32})^2$$

$$2^{64} = (\mathbf{68})^{\mathbf{2}}\ \boldsymbol{mod}\ \mathbf{101}$$

$$2^{64} = \mathbf{4624}\ \boldsymbol{mod}\ \mathbf{101}$$

$$2^{64} = \mathbf{79}\ \boldsymbol{mod}\ \mathbf{101}$$

If we were to square both sides again we would exceed 2^{79} therefore we terminate our execution here. We then write down our results:

2^1	2^2	2^4	2^8	2^{16}	2^{32}	2^{64}
2 mod 101	**4** mod 101	**16** mod 101	**54** mod 101	**88** mod 101	**68** mod 101	**79** mod 101

Now, write our original exponent (79) in binary:

$$79_2 = 1001111$$

$$79 = 64 + 8 + 4 + 2 + 1$$

Then, using our above table we can represent this as:

$$2^{79} = 2^{64+8+4+2+1}$$

$$= (2^{64})(2^8)(2^4)(2^2)(2^1)$$

$$= (79)(54)(16)(4)(2)\ mod\ 101$$

$$= 546048\ mod\ 101$$

$$= 42\ mod\ 101$$