Pros:
- Clearly defined layers and protocols.
- Very general design that could be applied to *any* network (defence, cash machines *etc.*), not just the internet.

Cons
- Often overly complex.
- The *Presentation* and *Session* layers have minimal functionality that could easily be subsumed into tl **Application** layer.

Pros
- Suited to network *programming* (particularly `java.net`, which is designed around TCP/IP).
- Simpler than the OSI model.
- Well suited to the internet.

Question 1

(a) State one example of a network for which the 7-layer OSI model may provide a more suitable description than the 5-layer TCP/IP model. [1 mark]

Cons
- Layers and protocols are not always clearly defined.
- For example, some security protocols sit 'between' layers – see Lecture 13.
- Lacks generality.

(b) How many unique IPv4 addresses are there in the range `129.11.192.0/22`?

前22位是网络位，后10位是主机位，2^10=1024

[1 mark]

(c) Does the address `129.11.195.212` belong to the range `129.11.192.0/22`? Explain your answer.

129.11.192.0/22前22位固定，其中192转换为2进制为110000|00且只有最后两位可变最大为11即3，所以第三bytes最大为195，最后一节最大为255，总范围为129.11.192.0-129.11.195.255，129.11.195.212属于这个范围 [2 marks]

(d) Expand the following IPv6 address to its full, 16-byte form.

展开为2001:02f9:0000:0000:0000:ffcc:0018:1234
IPv4是4组，IPv6是8组

`2001:2f9::ffcc:18:1234`

0063可以略为63
0000可以略为0
0:0:0:0可以略为::

When there are multiple runs of all–zero values, only the **longest** should be contracted using a double colon.

- If the runs are the same length, the **leftmost** is contracted.

[2 marks]

(e) For each of the following applications, state whether you would use TCP or UDP, and provide a brief justification in each case.

(i) An email client that connects to a server to upload or download messages.

(ii) An online chess game with one other opponent.

(iii) An online multi-player vehicle racing game.

They differ in:
- The information provided in their headers.
- The reliability of service.
- Performance.

There are two key protocols for the Transport layer:
- **UDP** : User Datagram Protocol.
- **TCP** : Transmission Control Protocol.

[6 marks]

(f) Someone is writing a multi-threaded web browser that makes multiple connections to the same web server. The hope is that by sending and receiving HTTP queries simultaneously, all of the data for a web page (text, images, movies *etc.*) could be downloaded simultaneously, speeding up the browser. Some key parts of the code are shown below.

```
1     public class HTTPThread extends Thread {
2        private int numThreads = 10;
3        private int port = 80;
4        private String hostname = ...; // (not shown)
5
6        public void run() {
7            Socket s = new Socket(hostname,port);
8            // ...  (remainder not shown)
9        }
```

```
10
11        public static void main(String[] args) {
12            for( int i=0; i<numThreads; i++ ) {
13                 HTTPThread t = new HTTPThread();
14                 t.start();
15            }
16            // (see question)
17        }
18    }
```

(i) Identify the two mistakes in the given code that will stop it from compiling. For each mistake, state the line number(s) where the problem lies, and briefly explain how you would rectify the mistake.

[4 marks]

(ii) Why is the port number fixed to be 80 in the code, rather than taking an optional value set by *e.g.* a command line argument?

[1 mark]

(iii) Suppose you wanted to add new code at line 16 that required all of the threads to have finished their tasks before continuing. How would you modify the existing code to achieve this synchronisation?

[2 marks]

(iv) Someone suggests replacing lines 12 to 15 with an Executor from the Java library `java.util.concurrent`. State one potential advantage of this choice.

[1 mark]

[Question 1 Total: 20 marks]

TCP

Connection-oriented protocol, *i.e.* maintains a persistent connection between the two hosts.

- Ensures **reliable data transfer**, possibly by requesting re-transmission of lost or corrupt segments.
- Also provides a degree of **congestion control**.

20-byte header with:

- Sequence and acknowledgement numbers.
- Bits used for maintaining the connection.
- Some specialist fields.

The remaining fields are the same as UDP.

The sequence and acknowledgement numbers are used to guarantee ordering, and to check for missed packets.

Uses of UDP

An important use of UDP is to access a DNS (Domain Name System) server, to map a readable internet address to an IP address.

- We will cover this next lecture.

Also useful for **real-time multi-media**, often wrapped into the **Real-time Transport Protocol** (RTP)

- Sits between Application and Transport layers.
- UDP packets are numbered such that receiver can determine if packets are missing.
- No correction or retransmission.
- Receiver can interpolate lost data.

TCP:Email requires reliable delivery of data (e.g., messages, attachments) in the correct order, with error checkingand retransmission.
TCP:Chess is a turn-based game where each move is critical and must be delivered reliably. Speed is less important than accuracy
UDP: eal-time games like racing require fast transmission of data with minimal latency.Occasional loss of data packets is acceptable, as new position updates will quickly follow. UDP is suitable because it s faster and does not introduce delay due to retransmission or ordering.

(1)The constructornew Socket(String host, int port)throws a checked exception:IOException. However, therun()method doesnotdeclare that it throws exceptions, nor is the exception caught.new Socket(hostname, port)这个构造函数可能会抛出IOException（这是受检异常）。但是你在run()方法中并没有处理这个异常（也没有用throws声明抛出），编译器会因此报错

```
public void run() {
    try {
        Socket s = new Socket(hostname, port);
        // 其他逻辑...
    } catch (IOException e) {
        e.printStackTrace(); // 适当处理
    }
}
```

numThreads定义但未使用，numThreads = 10被定义了，但没有用在main中的循环中（硬编码了 10）

(2)Port80is thedefault port for HTTP, so when connecting to a typical web server over standard HTTP (not HTTPS), this is the expected port. It's likely hardcoded here for simplicity, assuming the server being tested or contacted is running a standard web service.端口80是HTTP 协议的默认端口，也就是说，大多数网页服务器（Web Server）在没有特别指定端口时，就监听的是端口 80。这段代码可能是为了简单演示或默认访问标准网站而写的，因此直接用 80 是可以工作的。

(3)public static void main(String[] args) {

```
    List<HTTPThread> threads = new ArrayList<>();

    for (int i = 0; i < 10; i++) {
        HTTPThread t = new HTTPThread();
        threads.add(t);
        t.start();
    }
    // 等待所有线程完成
    for (HTTPThread t : threads) {
        try {
            t.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
    // 所有线程完成之后才会执行这里
    System.out.println("所有线程任务已完成，可以继续执行后续代码。");
}
```

(4)Using anExecutorallows for better thread management by reusing a pool of threads, reducing the overhead of thread creation and improving scalability and performance.使用Executor可以复用线程池，减少频繁创建和销毁线程的开销，提高程序的性能和可扩展性

Question 2

Figure 1 gives code for a simple 'Quote Of The Day' server that sends a randomly-selected quote to each client that contacts it. The corresponding client application that communicates with the server is also provided. Inspect both pieces of code, then answer the following questions.

(a) For each of the following statements, say whether they are true or false, and provide a brief explanation to justify your decision, giving line numbers from the code where relevant.

(i) The code uses TCP as implemented by `java.net`.

(ii) The protocol is for the client to send a zero-size message to the server, which returns a random quote to the same client.

(iii) Both applications use the reserved port number 8787.

(iv) The maximum quote length that the server can send is defined by `rSize`.

[4 marks]

(b) While the code seems to work correctly when both the server and client applications are running on the same machine, the client sometimes hangs when the applications are running on remote hosts. The only modification made to the client code was to initialise `serverAddr` with the server's address in line 8, and this is known to work correctly.

(i) Give the line in the given client code where you think the application hangs.

[1 mark]

(ii) Why do you think this problem occurs?

[2 marks]

(iii) It is suggested that you modify the code to wait for a response from the server for some period of time, and give an error message if this time is exceeded. Give pseudo-code for how you would implement this. You do not need to provide working Java code, but should use the correct class and method names from `java.net`. Only provide pseudo-code specific to this new feature.

[4 marks]

(iv) It is further suggested that the client should make multiple attempts to communicate with the server before giving an error message. Briefly describe how you would achieve this. Only discuss this new feature in your answer.

[3 marks]

(c) It is now decided to upgrade both the client and server applications to use multi-casting.

(i) Name one real-world application (as opposed to this 'Quote Of The Day' example) that might benefit from using multi-casting.

[1 mark]

(ii) Why did you choose this example?

[1 mark]

(iii) You are asked to alter the client application to use multi-casting. List the key steps that you would include, in the order they should be performed, using correct names from `java.net` wherever possible. Only list steps that relate specifically to multi-casting.

[3 marks]

(iv) What is the smallest change you could make to the server application to make it work with the new multi-casting client?

[1 mark]

[Question 2 Total: 20 marks]

(a)(i) False.UDP: DatagramSocket DatagramPacket TCP: Socket ServerSocket
(ii) True. 客户端构造一个长度为 0 的DatagramPacket并发送出去(见客户端第 10 行): DatagramPacket packet = new DatagramPacket(new byte[0], 0, serverAddr, 8787);服务器收到后不检查内容，而是直接返回一条随机 quote(见服务器第 10 行): String quote = quotes[(int)(Math.random()*quotes.length)];
(iii) False.是的，两端都使用了端口8787，但是：Port 8787 并不是保留端口（reserved port）。保留端口是指0–1023（即所谓的 well-known ports），而8787在 1024–65535 范围内，是用户可用的端口。服务器第 7 行。客户端第 10 行。
(iv) False.rSize是客户端用于接收数据的缓冲区大小（客户端第 6 行）：private static int rSize = 1000;但服务器发送的 quote 是由quote.getBytes()决定的长度（服务器第 11 行），并不受rSize限制。也就是说，如果服务器返回的 quote 超过客户端的rSize，接收的数据会被截断，但服务器仍可发送更长内容。所以：rSize仅影响客户端能接收的最大长度，而不是服务器能发送的最大长度。

(b)(i)socket.receive(rPack);
(ii) DatagramSocket.receive()是一个阻塞方法，当没有收到数据包时，它会无限期地等待。如果客户端在远程机器上，可能存在以下网络问题导致数据包无法返回：服务器的响应没有成功返回客户端（如被防火墙阻止、NAT问题、端口未开放等）;服务器未收到请求（客户端的出站 UDP 被丢弃）;网络丢包;所以客户端会一直卡在receive()，看似“挂起”。
(iii)

```
DatagramSocket socket = new DatagramSocket();

// 设置接收超时时间，例如 3000 毫秒（3 秒）
socket.setSoTimeout(3000);

try {
    socket.receive(rPack); // 等待响应，最多等待 3 秒
    // 正常处理数据
} catch (SocketTimeoutException e) {
    // 超时：没有收到服务器响应
    print("服务器未响应（超时）");
}
```

(iv)目标：多次尝试通信（如最多尝试 3 次），每次等待一定时间，如果仍失败则报错。1.设置一个重试次数上限，例如MAX_RETRIES = 3
使用一个循环进行多次尝试：每次发送请求;设置超时并尝试接收响应;如果成功接收到，立即退出循环;如果超时，记录失败，继续下一次尝试。3.如果超过最大次数仍然失败，打印错误信息

```
int MAX_RETRIES = 3;
int attempt = 0;
boolean success = false;

while (attempt < MAX_RETRIES && !success) {
    send(requestPacket);

    try {
        receive(responsePacket);
        success = true;
        // 处理响应
    } catch (SocketTimeoutException e) {
        attempt += 1;
        print("第 " + attempt + " 次尝试超时，重试中...");
    }
}

if (!success) {
    print("服务器无响应，多次尝试失败。");
}
```

Code for the 'Quote Of The Day' (QOTD) server application:

```
1 import java.net.*;
2 import java.io.*;
3
4 public class QOTDServer {
5   private static String[] quotes={"Quote1","Quote2","Quote3"};
6
7   public static void main(String[] args) throws IOException {
8     DatagramSocket socket = new DatagramSocket(8787);
9
10    while(true) {
11      DatagramPacket rPack=new DatagramPacket(new byte[10],10);
12      socket.receive(rPack);
13
14      String quote=quotes[(int)(Math.random()*quotes.length)];
15      DatagramPacket sPack=new DatagramPacket(quote.getBytes(),
              quote.length(),rPack.getAddress(),rPack.getPort());
16      socket.send(sPack);
17    }
18  }
19 }
```

Code for the 'Quote Of The Day' (QOTD) client application:

```
1 import java.net.*;
2 import java.io.*;
3
4 public class QOTDClient {
5   private static int rSize = 1000;
6
7   public static void main(String[] args) throws IOException {
8     InetAddress serverAddr = InetAddress.getLocalHost();
9     DatagramSocket socket = new DatagramSocket();
10
11    DatagramPacket sPack = new DatagramPacket(new byte[0],0,
                                              serverAddr,8787);
12    socket.send(sPack);
13
14    DatagramPacket rPack = new DatagramPacket(new byte[rSize],
                                                  rSize);
15    socket.receive(rPack);
16
17    if( rPack.getAddress().equals(serverAddr) )
18      System.out.println(new String(rPack.getData(),"UTF-8"));
19  }
20 }
```

Figure 1: Code for question 2.

Question 3

(a) To which layer in the protocol stack is ICMP most relevant?

Slide-2 p19 Network Layer [1 mark]

(b) Messages sent over the internet are typically broken down into packets.

(i) Suggest one advantage of doing this.

提高传输效率和可靠性 [1 mark]

(ii) Now suggest one disadvantage.

需要额外的重组和管理 [1 mark]

(c) A student is trying to develop their own transport protocol to send data reliably over an unreliable channel. So far they have implemented `rdt_send()` and `rdt_receive()` functions, where `rdt_receive()` sends an acknowledgment to the sender whenever a packet is received. When `rdt_send()` receives the acknowledgement it sends the next packet. If `rdt_send()` does not receive any acknowledgement after a predefined time-out, it resends the original packet. There are no sequence numbers in the protocol's header.

(i) If packets can be lost during transmission but are not corrupted, will the student's current implementation work? Explain your answer.

[2 marks]

(ii) The student now includes a checksum in the packet header, and `rdt_receive()` sends a negative acknowledgement to the sender if packet corruption is detected, which then re-sends the packet. The corrupted packet itself is dropped. The receiver still sends a positive acknowledgement if the packet was received uncorrupted. If packets cannot be lost but may be corrupted, will this new scheme work?

[2 marks]

(iii) Irrespective of its reliability, timing tests suggest that the performance of the new protocol is poor. Why is this? Suggest one modification you would make to improve performance.

[2 marks]

(d) Consider the network of five nodes `u`, `v`, `w`, `x` and `y` and their corresponding link costs shown in Fig. 2. Determine the optimal routing from node `u` for this network using Dijkstra's algorithm. To help you, the first row of the table of the set of nodes with known least cost path N', the current value of path cost D, and predecessor node along the path p, is as follows:

Step	N'	$D(v)$, $p(v)$	$D(w)$, $p(w)$	$D(x)$, $p(x)$	$D(y)$, $p(y)$
0	u	3,u	2,u	5,u	∞

Complete this table as per Dijkstra's algorithm, then answer the following questions.

(i) What is the step number in the final row of your table?

[1 mark]

(ii) Copy the column `N'` from your table into this box, starting from (and including) the entry u given above.

[4 marks]

(iii) What is the final value in the column $D(v)$, $p(v)$ on your table?

[1 mark]

(iv) What is the final value in the column $D(x)$, $p(x)$ on your table?

[1 mark]

(v) Finally, what is the final value in the column $D(y)$, $p(y)$ on your table?

[1 mark]

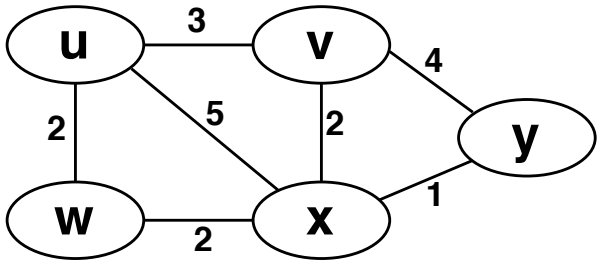


Figure 2: Figure for question 3(d).

(e) Give two reasons why Dijkstra's algorithm is typically only used for small sub-networks.

[2 marks]

(f) What class of algorithm could be considered instead for larger networks?

[1 mark]

[Question 3 Total: 20 marks]

[Grand Total: 60 marks]

Network layer

Most common protocol is the **Internet Protocol**, or IP.

- Describes how data is grouped into **datagrams** *(packets)*.
- The network **addressing scheme**.

A **datagram** is produced containing the data and header information.

Also ICMP (Internet Control Message Protocol), which is how routers communicate to ensure efficient transport of messages through the network.

Java only understands IP.

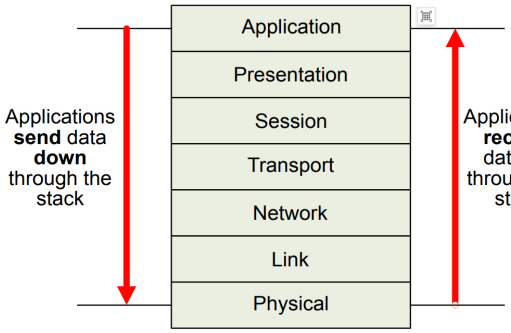
The TCP/IP 5-layer model

The most widely used mode for the internet, and the most relevan for this course, is a simplified version of the OSI model:

Application
Transport
Network
Link
Physical

- *Presentation* and *Session* layers merged into *Application*.
- *Network* layer sometimes called the *Internet* layer.
- (*Data-*)*Link* and *Physical* layers sometimes merged to give a 4-layer model (or the *Physical* layer simply dropped).

Layers in the OSI model



End

(c)(i)如果数据包可能丢失但不会被损坏，该协议是否能正常工作？答：不能完全可靠地工作。因为：学生的协议没有使用序列号（sequence numbers）；当rdt_receive()收到一个重复包（如前一个包的重传），它仍然会发送一个 ACK；如果 ACK 丢失，发送方在超时后会重发上一个数据包；接收方无法判断这个包是新的还是旧的（因为没有序列号），于是会重复交付数据给上层应用层，导致数据重复。

(ii)如果数据包不会丢失，但可能被损坏，该协议是否能正常工作？因为：现在协议加入了校验和（checksum），可以检测出数据包是否损坏；如果数据包损坏，接收方发送NAK（负确认），并丢弃数据包；如果数据包没有损坏，接收方发送ACK。由于数据不会丢失，只会损坏，所以：发送方一定能收到 ACK 或 NAK；收到 NAK 后会重发；接收方不会交付损坏的数据包。

(iii)性能测试显示新协议效率低，为什么？如何优化？因为：使用的是停等协议（Stop-and-Wait Protocol）。当前协议是“发送一个包 → 等 ACK → 再发送下一个包”的模式；即使网络没有丢包，每次发送都需要等待一个往返时间（RTT），这极大地限制了吞吐量Throughput；带宽利用率低Bandwidth Utilization，尤其在延迟高的网络中更明显。改进建议：使用流水线协议（Pipelining）。实现类似Go-Back-N或Selective Repeat协议：
发送方可以在不等待 ACK 的情况下，连续发送多个数据包；接收方根据序列号来接收和确认每一个包；ACK 和 NAK 会用于通知哪些包已成功/需要重传；保持窗口大小，控制并发包数Number of Concurrent Packets，提升带宽利用率。

Dijkstra's link-state routing algorithm

Assume topology and costs of entire (sub-)network known to all hosts, achieved by *i.e.* link state broadcast.

- Computes least cost paths from **source** node to all others.
- Gives **forwarding table** for that node.
- **Iterative**: After k iterations, knows least cost to k destinations.

Notation:

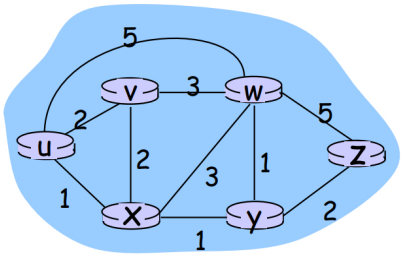
- $c(x, y)$ is link cost for $x \to y$, or ∞ if no direct link.
- $D(v)$ is current path cost from source to v.
- $p(v)$ is the predecessor node along path to v.
- N' is set of nodes whose least cost path is known.

(d)(i)4 (ii)uwvxy (iii)3, u (iv)4, w (v)5, x

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)
0	u	2,u	5,u	1,u	∞
1	uw	2,u		4,w	∞
2	uwv			4,w	7,v
3	uwvx				5,x
4	uwvxy				

Dijkstra's algorithm: Example

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



(e)原因说明
1.计算复杂度高算法在大规模网络中处理速度慢，不适合快速响应,High Computational Cost;Dijkstra’s algorithm hastime complexity of O(N²) in its basic form (orO((N + E) log N)with a priority queue);Inlarge networks, with thousands of routers and links, this becomes computationally expensive;Routers would need significant processing power and time to compute routes across the entire network.
2.拓扑变化频繁时重计算代价大每次变动都要重新运行算法，影响路由稳定性,Frequent Updates Due to Topology Changes;In large-scale networks (like the Internet),network topology changes frequently(e.g., due to link failures, congestion, or policy changes);Dijkstra’s algorithm must berecomputed from scratchwhen any link cost changes;For large networks, this would cause high overhead andslow convergence, which affects routing stability.

(f)Distance Vector algorithms, such as RIP.

Today's lecture

Todays lecture is the second of two on the Network layer and we will look at how the outgoing path for each packet is actually determined, *i.e.* **routing algorithms**:

- **Dijkstra's algorithm**, a link state algorithm.
- **RIP**, a distance vector algorithm.
- **OSPF** and the hierarchical version, that work within subnetworks.
- **BGP**, the *de facto* standard for routing between subnetworks.

We will only consider destination IP addresses, and assume the goal is to be transported 'efficiently.'

Routing algorithm classification

Global or **decentralised** information?

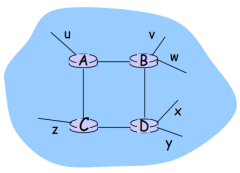
- If **global**, all routers have complete knowledge of topology and link costs. Leads to **link state** algorithms.
- If **decentralised**, each router knows only physically-connected routers and link costs to them.
  - **Iterative** process of computation and information exchange with neighbours (**distance vector** algorithms).

Can also be **static** or **dynamic**:

- **Static** suitable if routes change slowly.
- **Dynamic** if routes change more quickly, including link cost changes.

RIP: Routing Information Protocol

- **Distance vector** algorithm, an iterative algorithm that only knows about local links and connected nodes.
- Included in BSD-UNIX[1], the precursor to FreeBSD, OpenBSD *etc.*, in 1982.
- Distance metric is the **number of hops** (maximum of 15).



From router A to subsets:

destination	hops
u	1
v	2
w	2
x	3
y	3
z	2