Question 1

(a) Does the address range 212.58.64.0/18 correspond to IPv4 or IPv6?

IPv4

[1 mark]

(b) Does the address 212.58.126.201 belong to the range 212.58.64.0/18? Explain your answer, showing binary where relevant.

212.58.64.0→212.58.01000000.0→212.58.01111111.255→212.58.127.255
属于

[4 marks]

(c) How can the range 197.12.104.11/16 be expressed using the asterisk notation?

197.12.*.*

[1 mark]

(d) Does 197.12.104.11/16 correspond to a class in the classful model? If so, which? If not, explain why not.

class C: 192.0.0.0→223.255.255.255
197.12.104.11/16: 197.12.0.0→197.12.255.255 Yes.

[3 marks]

(e) Suppose a server is listening to port number 8765, and when a client connects, the `accept` method returns a socket that it uses to communicate with that client. The client application also has a socket that it uses to communicate with the server.

As part of a debugging exercise, both client and server output their local and destination ports for their respective sockets. What do you expect the port numbers to be, or what relationship do you expect them to obey?

[3 marks]

(f) The remaining subquestions in Question 1 refer to the code given in Figure 1, which is for a simple client application that accesses the server `dict.org`, and uses it to translate a series of English words into French, one at a time. Inspect the code, then answer the following questions.

What does the number on line 16 of the given code refer to? Which level of the TCP/IP protocol stack does it most closely relate to?

[2 marks]

(g) Both the input and output streams are buffered. Why is this normally desirable for networked applications?

[2 marks]

(h) You are told that `dict.org` is a popular translation service for phrases or single sentences. What multi-threaded architecture do you think is most suitable for this application? Explain your reasoning making specific reference to this particular application.

[4 marks]

[Question 1 Total: 20 marks]

Code for the 'Network Translator' client application.

```
1  import java.io.*;
2  import java.net.*;
3
4  public class NetworkTranslator
5  {
6    private Socket socket = null;
7    private BufferedWriter writer = null;
8    private BufferedReader reader = null;
9
10   public static void main( String[] args ) {
11     NetworkTranslator translator = new NetworkTranslator();
12     translator.translateWords( args );
13   }
14
15   public void translateWords( String[] words ) {
16     socket = new Socket("dict.org",2628);
17
18     try {
19       writer = new BufferedWriter(
20                  new OutputStreamWriter(
21                    socket.getOutputStream(),"UTF-8"));
22
23       reader = new BufferedReader(
24                  new InputStreamReader(
25                    socket.getInputStream(),"UTF-8") );
26
27       for( String word : words ) {
28         // Send the request to translate a word.
29         writer.write( "DEFINE fd-eng-fra " + word + "\r\n" );
30         writer.flush();
31
32         // Print each line returned by the server.
33         String ln;
34         for(ln=reader.readLine();ln!=null;ln=reader.readLine()) {
35           if(ln.startsWith("250 ")) break; // Last line.
36           if(ln.startsWith("552 ")) break; // Word not found.
37           System.out.println(ln);
38         }
39       }
40     }
41     catch( IOException ex ) {
42       System.err.println(ex);
43     }
44   }
45 }
```

Figure 1: Code for questions 1(f), 1(g) and 1(h).

Classful addressing

Originally there were several **classes** of IPv4 address:

- Class A: `0.*.*.*` to `127.*.*.*`
- Class B: `128.*.*.*` to `191.*.*.*`
- Class C: `192.*.*.*` to `223.*.*.*`
- Class D: `224.*.*.*` to `239.*.*.*`
- Class E: `240.*.*.*` to `255.*.*.*`

where the '`*`' means 'any value.'

They vary in size: Class A > Class B > Class C > Class D, E.

Problems with the classful model

This so-called **classful** model is not perfect.

Various classes are reserved, reducing space.

- Class D: *Multi-casting (Lecture 15).*
- Class E: Reserved for some unspecified 'future use.'
- Some special addresses:
  - `0.0.0.0` for 'this' machine.
  - `255.255.255.255` is used for **broadcasting** (device discovery).
  - `127.*.*.*` is used for *loopback*.

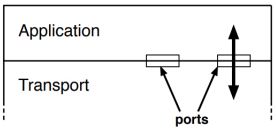
The least significant 3 bytes were not always managed efficiently.

(e) The client's local port is a random ephemeral port, and its destination port is 8765. The server's local port is 8765, and its destination port is the client's ephemeral port. These values form asymmetric connection, where each endpoint sees the other's port.
(f) It's the server's service port and belongs to the Transport Layer of the TCP/IP stack. 是远程服务器的服务端口，属于TCP/IP 的传输层
(g) To improve performance, reduce I/O calls, and facilitate line-based reading of text protocols. 提高性能、减少 I/O 次数、方便逐行读取文本协议
(h) Thread-per-connection— suitable for moderate concurrency, independent requests, and simple processing logic. 每连接一个线程（Thread-per-connection），适合中等并发、请求独立、处理简单的服务

Ports

Applications are linked to the Transport layer *via* **network ports**.

- Different port numbers can be associated with different Application layer processes.
- This allows multiple applications to run simultaneously on the same host with the same IP address.
- Purely software, provided by the OS.
- Can be allocated to a particular service, *e.g.* email, HTTP *etc.*



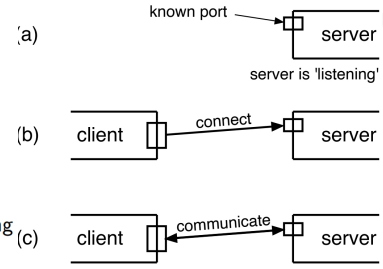
Sending and receiving

Converting host-to-host delivery to process-to-process delivery is the job of the Transport layer.

- Known as **multiplexing** (sending) and **demultiplexing** (receiving).

Applications typically publish the port they are **listening to** (receiving on).

- A sending process will attempt to establish a connection using the published port number.
- Will continue to communicate using the same port number, but the port can also be used to initiate new contacts.



Available ports

Ports 1-65535 are available on any given host.

- 16-bit unsigned int, with zero not allowed.
- TCP and UDP ports and **independent**.

Ports are characterised by:

- Ports 1-1023 are **reserved**; approved by IANA (Internet Assigned Numbers Authority).
- Commonly used ports lie outside this range, *i.e.* 1024-65535
- We can use any freely available ports in this range.

To see ports on a UNIX machine, look at `/etc/services`

Remote login and file transfer

`telnet`

- Port 23.
- Short for teletype network.
- The original client-server communication protocol.
- Now rarely used as it is insecure.

`ssh`

- Port 22.
- Secure shell.

`ftp`

- Stands for file transfer protocol.
- Port 21 for commands (`dir`, `put`, `get`, *etc.*)
- Port 20 used for data (the original use; slow).

Email

`smtp`

- Port 25.
- Stands for simple mail transfer protocol.
- Used for sending mail.

`imap`

- Port 143.
- Used to access mail.
- Stands for internet message access protocol.

`pop3`

- Port 110.
- Also used to access mail.
- Stands for post office protocol.

Web

`http`

- Port 80.
- Stands for hypertext transfer protocol.
- Insecure.

`https`

- Port 443.
- Secure version of `http`, requiring authentication.
- We will look at security and authentication in Lecture 1

Altogether

Link layer header (*e.g.* 22 bytes for Ethernet)
Network layer header (20 or 40 bytes)
Transport layer header (8 or 20 bytes)
Message . . .

Note each header has a source and destination **'address'**:

- **Ports** in between the Transport and Application layers.
- **IP Addresses** in the Network layer.
- **MAC Addresses** in the Link layer.

Summary

- Headers are prepended to data at most levels as it is being sent, and removed at the same level as it is being received.
- Naming and addressing is key.
  - **MAC addressing** at the Link layer (not relevant to programming).
  - **IP addressing** at the Network layer, allows high-level specification of source and destination.
  - **Port numbering** at the Transport layer allows specific referencing to applications.

Next time we will look at DNS, the Domain Name System.

Question 2

(a) State whether the following statements are true or false. For each of those you think are false, provide a reason to justify your answer.

(i) If a UDP server application tries to bind to a port number that is already is use in a TCP server application, an error will occur.

(ii) The real-time transport protocol RTP uses UDP for speed.

(iii) A server utilising non-blocking I/O is likely to outperform a multi-threaded server in situations where clients typically connect for a long time and frequently exchange messages with the server.

(iv) Session keys are normally encrypted using asymmetric algorithms because they are faster than symmetric algorithms.

(v) Authentication is the problem of confirming the sender of a message really is who they claim to be.

[8 marks]

(b) The remaining subquestions in Question 2 refer to the following situation. A local river authority has installed monitoring equipment at a single site on a river which measures the height of the water at 60 second intervals. They would like this measurement to be sent out to multiple parties over the internet, and ask you to develop a multicast server in Java to do this.

Firstly, what are the requirements for the IP address and port number that the server must use for multicast communication? Give your answer for both IPv4 and IPv6.

[3 marks]

(c) When initialising the server socket, should you use `ServerSocket` from `java.net`? If not what should you use instead?

[2 marks]

(d) Provide pseudo-code for a minimal implementation of the loop that sends the water height. You are provided with a function `byte[] getWaterHeight()` that returns the current height in a format suitable for sending. You should use the correct names from `java.net` and `java.io`, but do not need to write full working code. Assume that the socket object has been initialised and is called `socket`. You do not need to include exceptions or error handling in your answer. Marks will be lost for code that is not relevant to this application.

[7 marks]

[Question 2 Total: 20 marks]

(a)(i) False. UDP and TCP areseparate transport protocols, and they maintainindependent port spaces. So, a UDP and a TCP server can both bind to the same port number on the same machinewithout conflict (e.g., TCP port 80 and UDP port 80 can coexist).

(ii) True. RTP (Real-time Transport Protocol) is designed for real-time applications like audio/video streaming, where speed and low latencyare more important than reliability. It typically runs over UDP because UDP avoids the overhead of connection setup, reliability, and congestion control found in TCP.

(iii) False. Non-blocking I/O (e.g. Java NIO) is well-suited formany clients with occasional activity, not for client thatfrequently exchange messages. If clients communicatefrequently and stay connected for long durations, amulti-threaded model(e.g., thread-per-connection or thread pool) can provide better throughput and lower latency due todedicated resourcesper connection.

(iv) False. Symmetric algorithms are much fasterthan asymmetric ones. That's why session keys (used for bulk encryption) are generated, then encrypted using an asymmetric algorithm(like RSA) during key exchange. After that, all data is encrypted using the faster symmetric key (e.g., AES).

(v) True. Authentication is exactly that — verifying the identity of a message sender, often using mechanisms like digital signatures, certificates, or credentials.

(b) Use239.0.0.0/8(IPv4) orff0e::/16(IPv6) multicast addresses with a port ≥1024
Multicast IP range:224.0.0.0to239.255.255.255 (as defined by RFC 5771)Recommended practice:Use addresses from the administratively scoped range:239.0.0.0/8(e.g.,239.1.2.3)Port number:Must be in the range1024–65535(avoid reserved ports <1024);Should be the same for sender and all receivers Example IPv4 Multicast Group:239.1.2.3:5000
Multicast IP range:Begins withff00::/8 The second nibble defines thescope, e.g.:ff02::- link-loca;lff05::- site-local;ff0e::- global scope. Choose based on reach:For wider Internet multicast:ff0e::1 Port number:Same rules as IPv4 (1024–65535) Example IPv6 Multicast Group:ff0e::1234:5678on port5000

(c) No, you shouldnotuseServerSocket. ServerSocketis used forTCP servers, which require connection-oriented communication. Multicast uses UDP, which is connectionless. Instead, use:MulticastSocketfromjava.net This class supportsjoining multicast groups and sending/receiving datagramsto/from multiple receivers.

(d)

```
InetAddress group = InetAddress.getByName("239.1.2.3"); // Example multicast group
int port = 5000;

while (true) {
    byte[] data = getWaterHeight();  // Get measurement
    DatagramPacket packet = new DatagramPacket(data, data.length, group, port);
    socket.send(packet);  // Send to multicast group
    Thread.sleep(60000);  // Wait 60 seconds before next reading
}
```

Question 3

(a) Why is it so difficult to change Network layer protocols, such as from IPv4 to IPv6, but comparatively easy to change Link layer protocols, or Application layer protocols?

[2 marks]

(b) A student is attempting to devise a flow control system over a reliable channel, that is, a communication channel over which packets are not lost or corrupted. Their idea is that the **sender** first sends a `start` message to the **receiver**, and the **receiver** returns the maximum message size `RcvWindow` it can currently receive. The **sender** then sends the message in packets that are no larger than `RcvWindow`, receiving from the **receiver** its new `RcvWindow` each time. Once the entire message has been sent, the **sender** sends a `stop` message to the **receiver**, which ends the communication. Figure 2 gives the student's first attempt at a finite state machine for the **sender**. Inspect the diagram, and then answer the following questions.

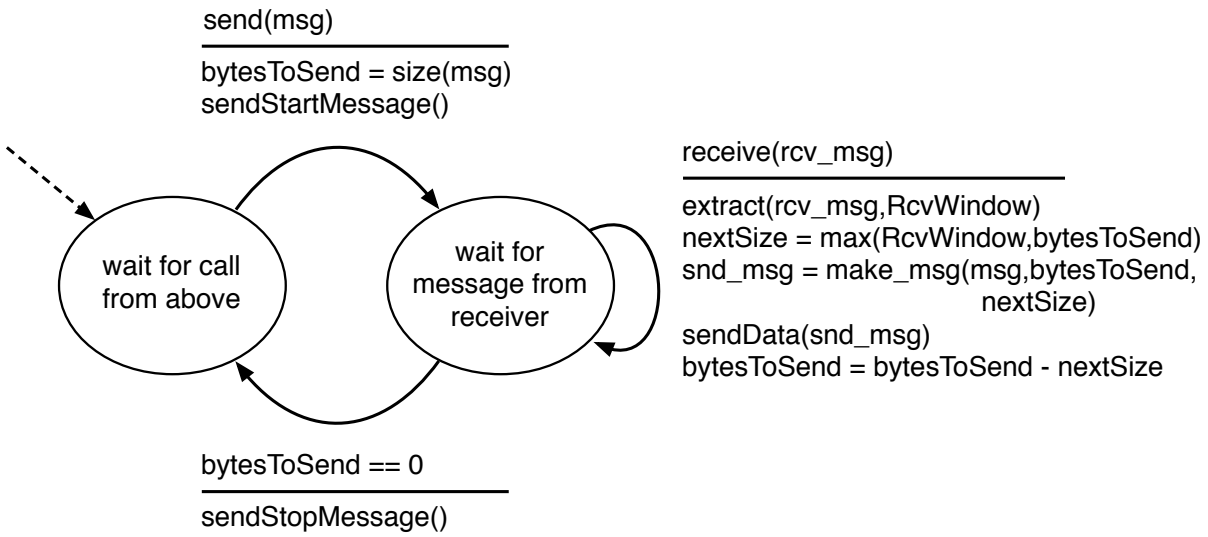


Figure 2: Finite state machine for a **sender** with flow control. Corresponds to question 3(b).

Identify one mistake in the **sender** finite state machine in Figure 2, and explain what you would do to correct the mistake.

[2 marks]

(c) Figure 3 shows the beginnings of the finite state machine for the **receiver**. Complete the diagram using the same notation as Figure 2, and describe the changes you made in the following subquestions. You may assume a function `sendRcvWindow()` is available, which sends the current `RcvWindow` to the **sender**. In addition, you may use the functions `recvStartMessage()`, which corresponds to the **sender**'s `sendStartMessage()`, and `recvStopMessage()` corresponding to the **sender**'s `sendStopMessage()`. The purpose of any other functions you use should be obvious from their name(s). You should not need to add any additional states.

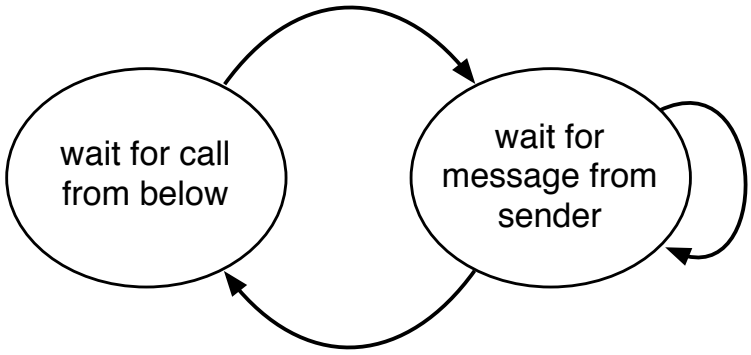


Figure 3: Starting point for the receiver with flow control. Corresponds to question 3(c).

Firstly, what have you put next to the arrow from 'wait for call from below' to 'wait for message from sender'?

[2 marks]

(d) What have you put next to the arrow from 'wait for message from sender' to itself?

[3 marks]

(e) What have you put next to the arrow from 'wait for message from sender' to 'wait for call from below'?

[2 marks]

(f) Finally, what other changes did you make to the diagram, if any?

[1 mark]

(g) The remaining questions refer to the configuration of four rourers A, B, C and D, and four destination networks w, x, y and z, given in Figure 4. The initial routing table for A is:

Destination	Next router	No. hops
w	-	1
x	D	4
y	D	3
z	D	2

Firstly, is RIP a global or decentralised algorithm? In which layer in the protocol stack does it primarily function?

[2 marks]

(h) State one advantage and one disadvantage of Dijkstra's algorithm compared to RIP.

[2 marks]

(i) Suppose B sends the following advertisement to A:

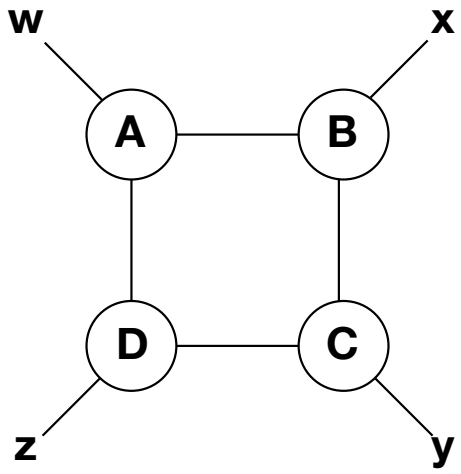


Figure 4: Corresponds to question 3(g), 3(h) and 3(i). A, B, C and D are nodes; w, z, y and z are subnetworks.

Destination	Next router	No. hops
w	A	2
x	-	1
y	C	2
z	C	3

What is the resulting routing table for A?

[4 marks]

[Question 3 Total: 20 marks]

[Grand Total: 60 marks]

End

(a) It is difficult to change Network layer protocols like IPv4 because they require changes across all routers and hosts on the Internet. In contrast, Link layer protocols are local to each network, and Application layer protocols are modular and easy to update in software.

Transition from IPv4 to IPv6

Not all routers can be upgraded simultaneously.

- *i.e.* there is no 'flag day' when the entire internet would switch to IPv6.

Therefore IPv4 and IPv6 must co-exist, for some time at least.

- How can the network operate with mixed IPv4 and IPv6 routers?

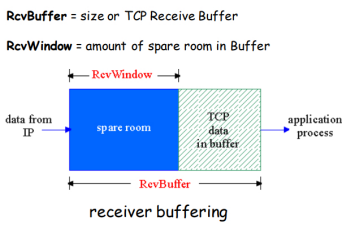
Tunnelling:

- IPv6 carried as **payload** in IPv4 datagram among IPv4 routers.

TCP Flow Control

The idea of **flow control** is that the sender won't overrun receiver's buffer by transmitting too much, too fast.

- **Receiver:** Explicitly informs the sender the (dynamically changing) amount of free buffer space.
  - The `RcvWindow` field in TCP segment.
- **Sender:** Keeps the amount of transmitted, unacknowledged data less than most recently received `RcvWindow`.



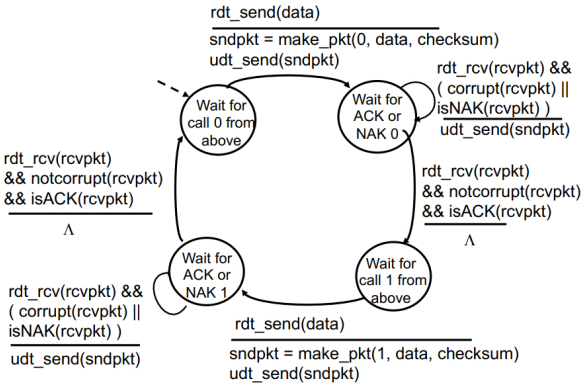
receiver buffering

(b) No handling of RcvWindow = 0 (zero window size)Correction: Add a condition for zero-window size
In flow control,RcvWindow(receiver's advertised window) indicates how much data the receiver can accept.The sender's FSM assumesnextSize = max(RcvWindow, bytesToSend), but this is problematic ifRcvWindow == 0.如果RcvWindow == 0，而bytesToSend > 0，那么nextSize = bytesToSend,meaning the sender wouldignore the receiver's zero windowand try to send data anyway, which violates flow control principles.There'sno mechanism in the FSM to waitor pause when the window is zero. It just keeps looping.
将max()改为min()，并在执行发送前添加对RcvWindow > 0的判断，即：

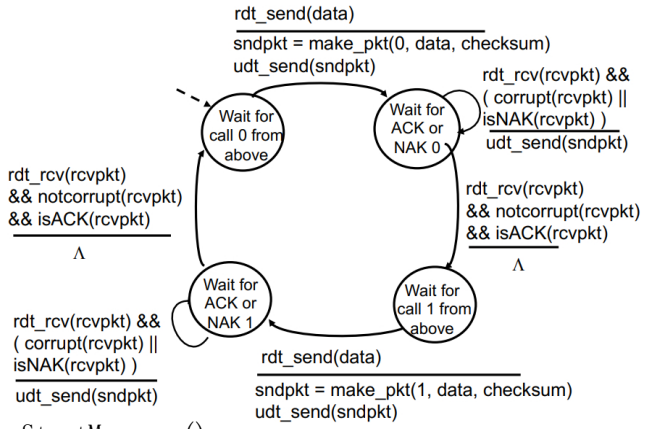
```
extract(rcv_msg, RcvWindow)

if (RcvWindow > 0) {
    nextSize = min(RcvWindow, bytesToSend)
    snd_msg = make_msg(msg, bytesToSend, nextSize)
    sendData(snd_msg)
    bytesToSend = bytesToSend - nextSize
}
```

(c)

Sender FSM with sequence numbers 0, 1



Sender FSM with sequence numbers 0, 1



Corrupted acknowledgements

However, this scheme has a fatal flaw — the **ACK** / **NAK** messages can **themselves become corrupted**!

- If the sender detects an error in an acknowledgement, it can re-send the packet.
- But the receiver does not know its acknowledgement was corrupted, and would interpret this as the **next** packet.
- We have **duplicate packets**.

Can solve this by using **sequence numbers**, which is exactly what TCP does — see TCP header structure on an earlier slide.

(c) recvStartMessage()
sendRcvWindow()
(d) receiveData()
sendRcvWindow()
(e) recvStopMessage()
sendRcvWindow()
(f) I added a self-loop on the 'wait for message from sender' state, triggered byreceiveData()and followed bysendRcvWindow(). This models the receiver handling ongoing incoming data while updating the sender with the current receive window. No new states were added.我在 "wait for message from sender" 状态上增加了一个自环，当收到数据时触发receiveData()，接着发送sendRcvWindow()。这个改动用于表示接收方在传输过程中持续接收数据并反馈其接收窗口状态。除此之外没有新增状态。

RIP Advertisements

Distributed algorithm that communicates **asynchronously** with directly connected nodes.

Distance vectors are exchanged among neighbours every 30 seconds *via* a **response message**, also known as an **advertisement**.

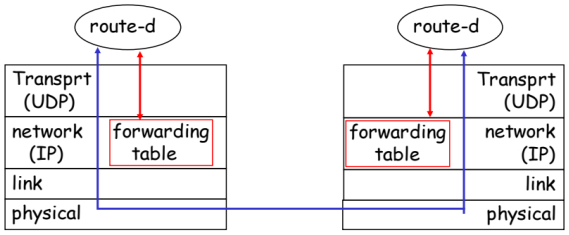
Each **advertisement** is list of up to 25 destination networks with the subnetwork.

- The **costs**, defined as the number of **hops**.
- Smaller messages that Dijkstra's algorithm.
- Tends to have slower convergence, with no guarantees.

RIP Table Processing

- RIP routing tables are managed by Application layer processes called **route-d** ('daemon').
- Advertisements are sent in UDP packets, periodically repeated.

Note this technically **breaks** the layered network architecture.



(g)RIP 是一个decentralised（分布式）算法that only exchanges information with directly connected routers using distance vectors。RIP 主要工作在Application Layer（应用层），even though it manages routing, because its messages are processed by an application-level daemon and sent via UDP.
(h)Advantage:Dijkstra converges faster and produces optimal paths due to full topology knowledge.
Disadvantage:Dijkstra has higher complexity and communication overhead due to global state maintenance.

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.

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.

(i)判断路由表属于哪个路由器的方法：
1.看哪一个目标是“直连”（跳数为 1 且 Next Router 是 “-”）
如果某条目的Next Router 是 “-” 或空白，且跳数是1，那说明该路由器直接连接这个网络。
所以你可以通过直连网络反推出这张表是谁的。
例如：

Destination	Next Router	No. hops
x	-	1

说明该路由器直连 x，也就是说这个路由器就是连接 x 的那台（比如 B）
多个目的网络都通过同一个 Next Router，说明这是另一个方向,如果某张表里到很多目的地都通过比如 C → 那么说明这张表不是 C 的，而是 C 的邻居的。

Destination	Next Router	No. hops
x	B	2

要到达网络 x，请把数据包发给路由器 B，这条路径总共需要 2 跳
解题过程：
A 和 B 是直接相连的，A 到 B = 1 hop，所以 A 经由 B 到达每个目的地的路径成本为：

Destination	Next	B 的跳数	A 通过 B 的跳数 = B 的跳数 + 1
w		2	3
x		1	2
y		2	3
z		3	4

与 A 原路由表比较，判断是否更新

Destination	Next	A 原跳数（via D）	B 广播路径（via B）更新？
w		1	3 不，1比3短
x		4	2 更，4比2长
y		3	3 不，一样
z		2	4 不，一样

A的最终路由表为：

Destination	Next Router	No. hops
w	-	1
x	B	2
y	D	3
y	D	2

Q1 B 向 A 发送路由通告时，为什么我们用的是“从 A 到 B 是 1 跳”？而不是“从 B 到 A”？是不是一样？
答案：是一样的！因为链路是双向的，跳数对称。在 RIP 中，所有链路默认被认为是对称且等价的：如果 AB 相连，那么A 到 B 是 1 跳，B 到 A 也是 1 跳。
Q2 为什么我们在计算中说“从 A 到某目的地，经 B 需要跳数 = B 广播的跳数 + 1”？
B 给 A 发通告时告诉它：我 B 到 x 需要 1 跳
A 想知道：我如果要通过 B 到 x，我得先到 B（1 跳）再从 B 到 x（再加 B 给出的跳数），所以计算是：
A 经 B 到 x 的跳数 = 1（A 到 B）+ B 到 x 的跳数，这就解释了，“B 发广告给 A，为什么不是 B 到 A？”
其实我们是从A 的视角在更新自己的路由表，所以计算的是“A 到 x 经 B”的跳数。
Q3 A 和 B 是不是直连，是从哪里看出来的？
是的，是从图上判断出来的！
Q4 如果 A 和 B 不是直连，那跳数怎么算？
如果 A 和 B 之间有 1 个中间节点，比如 A-D-B　　那 A 到 B = 2 hops
如果是 A-D-C-B　　A 到 B = 3 hops
A 经由 B 到 x = A 到 B 的跳数 + B 到 x 的跳数