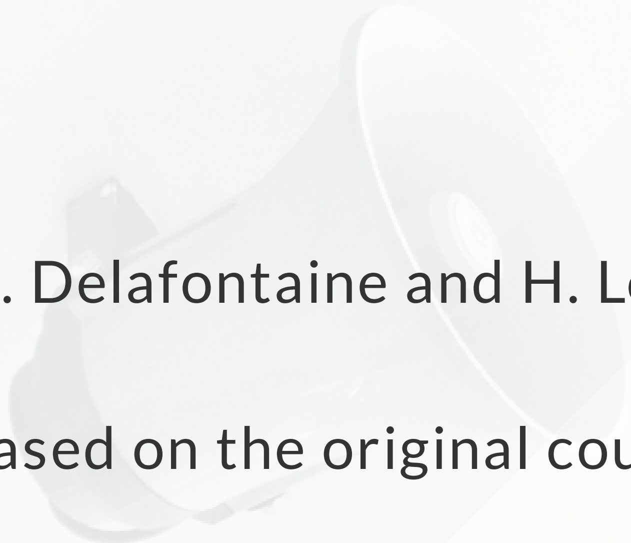


Java UDP programming

https://github.com/heig-vd-dai-course

Web · PDF

L. Delafontaine and H. Louis, with the help of GitHub Copilot.

Based on the original course by O. Liechti and J. Ehrensberger.

Objectives

- Learn the differences between TCP and UDP and reliability
- Learn what an UDP datagram is
- Learn the different ways to send a UDP datagram to one or multiple clients
- Learn UDP in the Socket API
- How UDP can be used for service discovery

Explore the code examples

More details for this section in the course material. You can find other resources and alternatives as well.

Explore the code examples

Individually, or in pair/group, **take 10 minutes to explore and discuss the code examples**.

Answer the questions available in the course material:

- How do the code examples work?
- What are the main takeaways of the code examples?
- What are the main differences between the code examples?

If needed, use the theoretical content to help you.

UDP

More details for this section in the course material. You can find other resources and alternatives as well.

UDP

- A transport layer protocol just like TCP
- Connectionless protocol - does not require to establish a connection before sending data
- Unreliable protocol - does not guarantee delivery but is fast
- Analogy: sending postcards through the postal service

Differences between TCP and UDP

More details for this section in the course material. You can find other resources and alternatives as well.

Differences between TCP and UDP

- TCP
  - Connection-oriented
  - Reliable
  - Stream protocol
  - Unicast
  - Request-response
  - Used for FTP, HTTP, SMTP, SSH, etc.

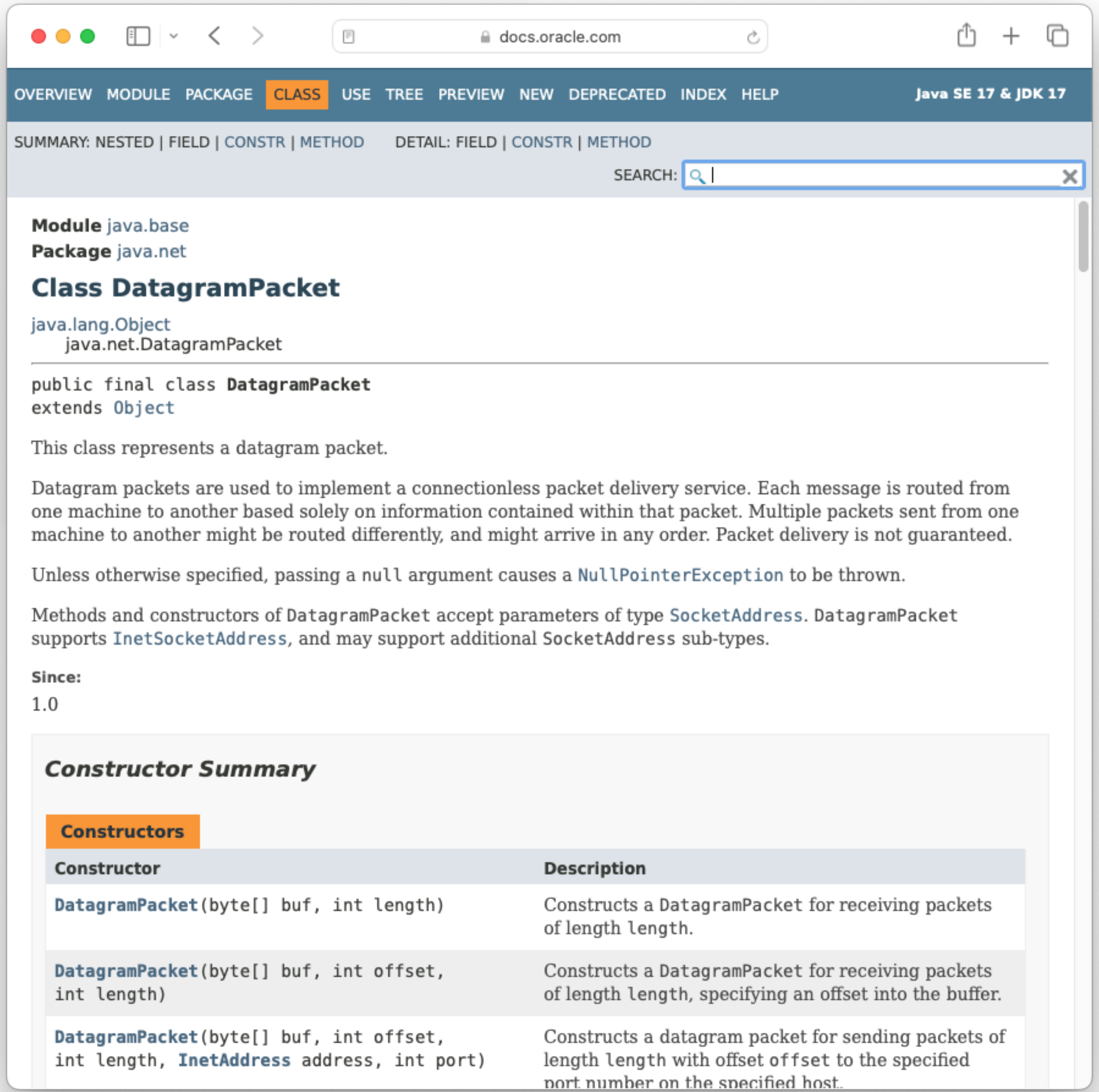
- UDP
  - Connectionless
  - Unreliable
  - Datagram protocol
  - Unicast, broadcast and multicast
  - Fire-and-forget, request-response (manual)
  - Service discovery protocols
  - Used for DNS, streaming, gaming, etc.

UDP datagrams

More details for this section in the course material. You can find other resources and alternatives as well.

UDP datagrams

- Datagrams = discrete chunks of data (packets) sent over the network
- Sent individually and independently
- Contain a header (source and destination ports, length, checksum, etc.) and a payload (data)


docs.oracle.com
OVERVIEW MODULE PACKAGE CLASS USE TREE PREVIEW NEW DEPRECATED INDEX HELP
Java SE 17 & JDK 17
SUMMARY: NESTED | FIELD | CONSTR | METHOD
DETAIL: FIELD | CONSTR | METHOD
SEARCH:

Module java.base
Package java.net

Class DatagramPacket

java.lang.Object
    java.net.DatagramPacket

```
public final class DatagramPacket
extends Object
```

This class represents a datagram packet.

Datagram packets are used to implement a connectionless packet delivery service. Each message is routed from one machine to another based solely on information contained within that packet. Multiple packets sent from one machine to another might be routed differently, and might arrive in any order. Packet delivery is not guaranteed.

Unless otherwise specified, passing a `null` argument causes a `NullPointerException` to be thrown.

Methods and constructors of `DatagramPacket` accept parameters of type `SocketAddress`. `DatagramPacket` supports `InetSocketAddress`, and may support additional `SocketAddress` sub-types.

Since:
1.0

Constructor Summary

Constructors

Constructor	Description
`DatagramPacket(byte[] buf, int length)`	Constructs a `DatagramPacket` for receiving packets of length `length`.
`DatagramPacket(byte[] buf, int offset, int length)`	Constructs a `DatagramPacket` for receiving packets of length `length`, specifying an offset into the buffer.
`DatagramPacket(byte[] buf, int offset, int length, InetAddress address, int port)`	Constructs a datagram packet for sending packets of length `length` with offset `offset` to the specified port number on the specified host.

Reliability

More details for this section in the course material. You can find other resources and alternatives as well.

Reliability

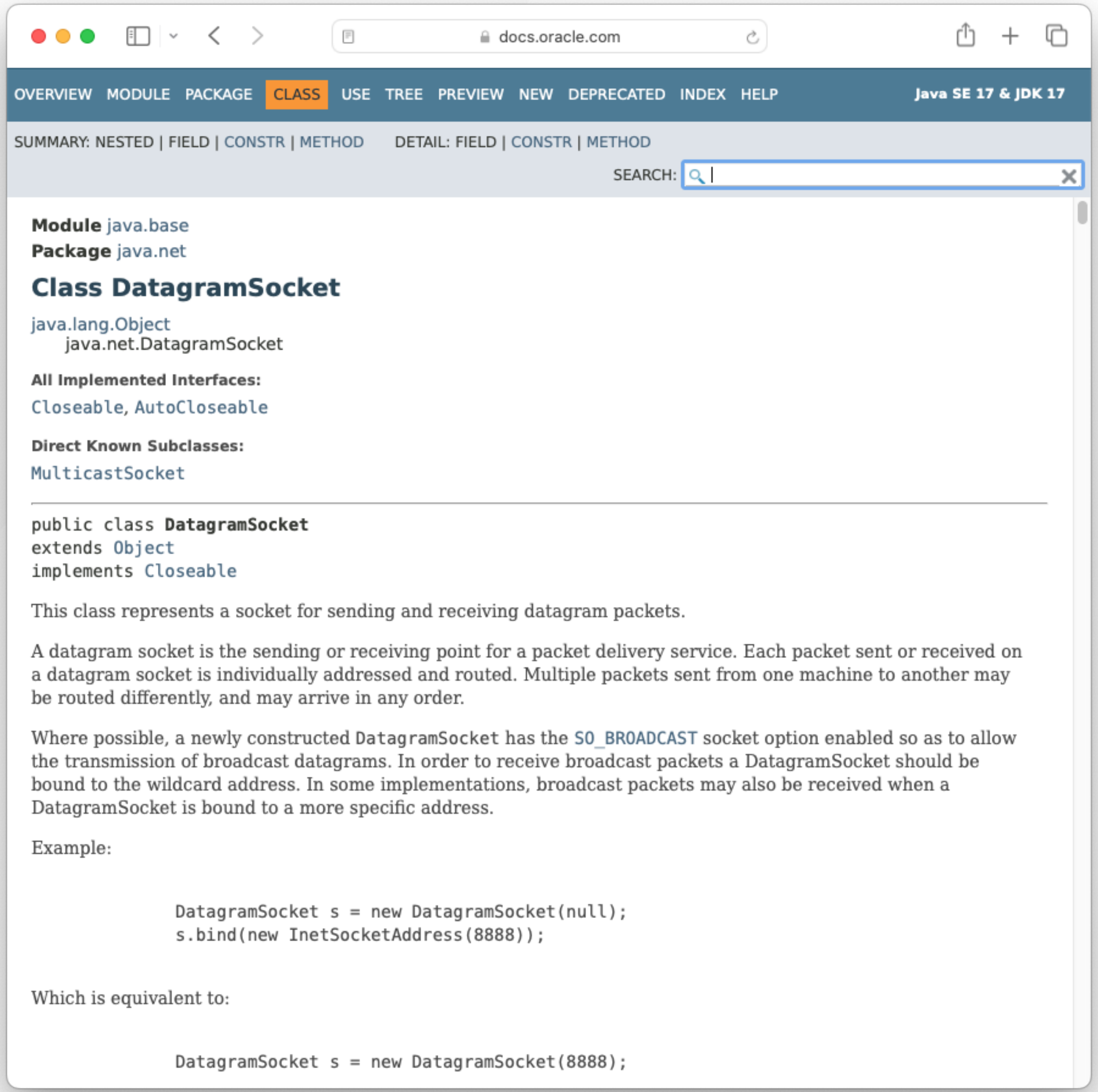
- UDP is unreliable (no guarantee of delivery, no guarantee of order)
- The application must implement its own reliability mechanism
- In some cases, reliability is not needed (e.g. streaming)
- Handling reliability is complex - not covered in this course

UDP in the Socket API

More details for this section in the course material. You can find other resources and alternatives as well.

UDP in the Socket API

- `DatagramSocket` is used to send and receive datagrams
- A datagram is created with the `DatagramPacket` class
- A multicast socket is created with the `MulticastSocket` class.


docs.oracle.com
OVERVIEW MODULE PACKAGE CLASS USE TREE PREVIEW NEW DEPRECATED INDEX HELP
Java SE 17 & JDK 17
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD
SEARCH:

Module java.base
Package java.net

Class DatagramSocket

java.lang.Object
  java.net.DatagramSocket

All Implemented Interfaces:
Closeable, AutoCloseable

Direct Known Subclasses:
MulticastSocket

```
public class DatagramSocket
extends Object
implements Closeable
```

This class represents a socket for sending and receiving datagram packets.

A datagram socket is the sending or receiving point for a packet delivery service. Each packet sent or received on a datagram socket is individually addressed and routed. Multiple packets sent from one machine to another may be routed differently, and may arrive in any order.

Where possible, a newly constructed DatagramSocket has the SO_BROADCAST socket option enabled so as to allow the transmission of broadcast datagrams. In order to receive broadcast packets a DatagramSocket should be bound to the wildcard address. In some implementations, broadcast packets may also be received when a DatagramSocket is bound to a more specific address.

Example:

```
DatagramSocket s = new DatagramSocket(null);
s.bind(new InetSocketAddress(8888));
```

Which is equivalent to:

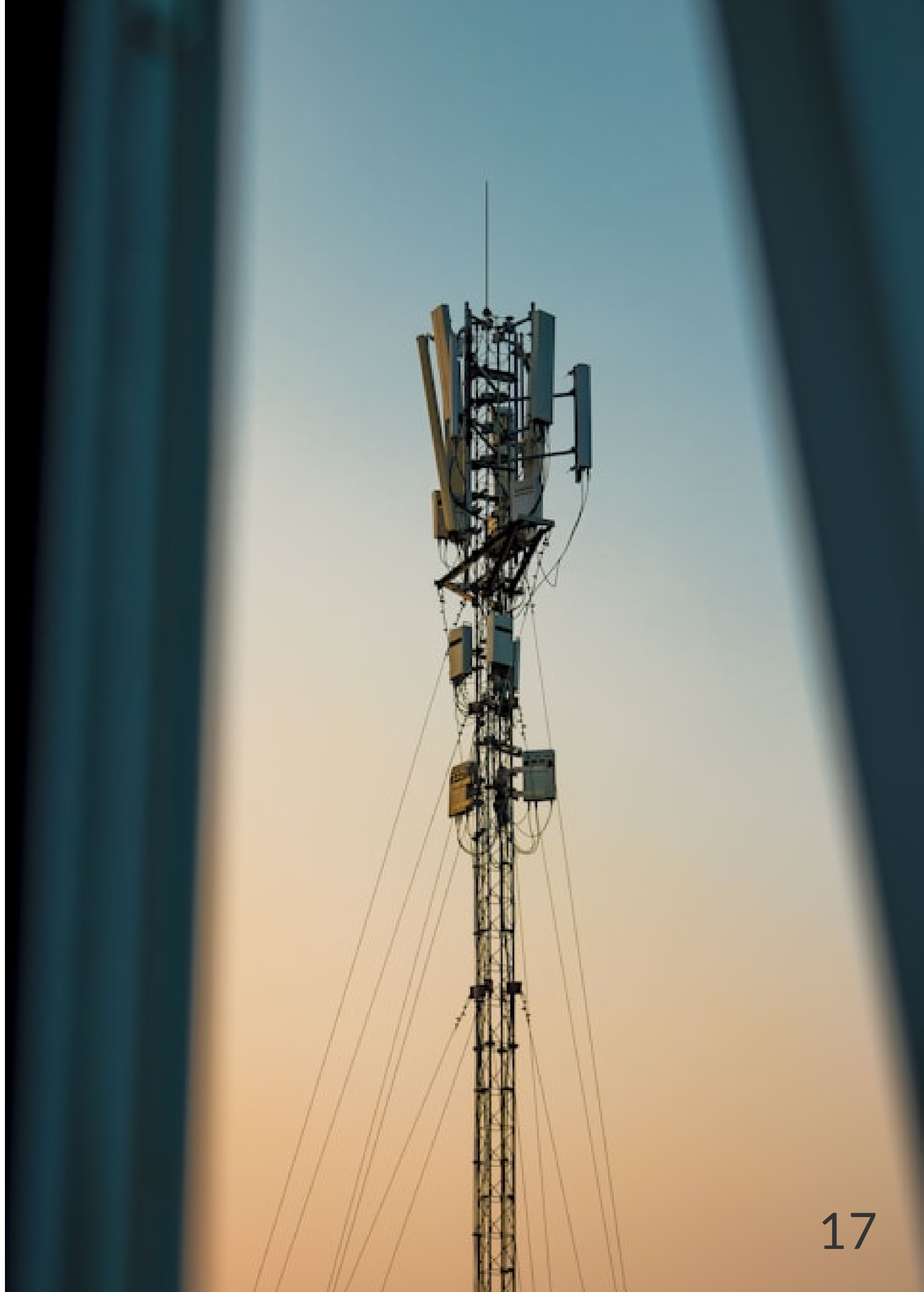
```
DatagramSocket s = new DatagramSocket(8888);
```

Unicast, broadcast and multicast

More details for this section in the course material. You can find other resources and alternatives as well.

Unicast, broadcast and multicast

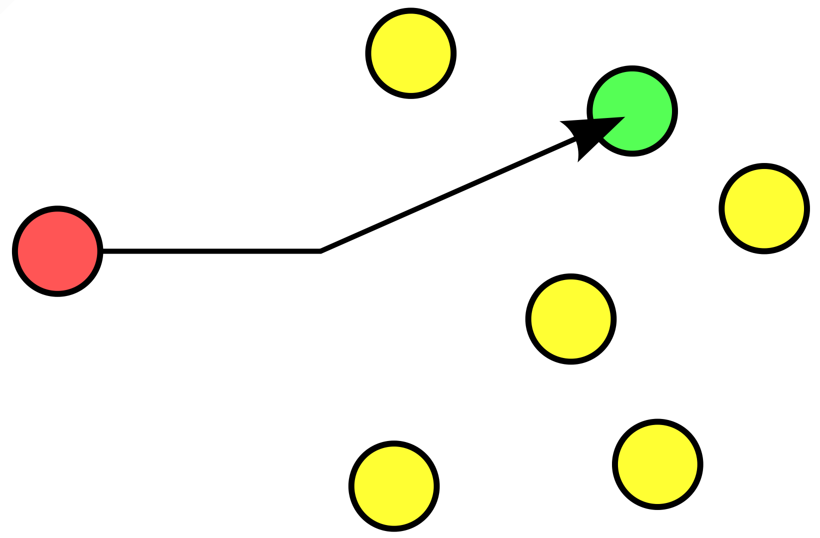
- Unicast, broadcast and multicast are ways to send data over the network
- TCP is unicast only
- UDP can be unicast, broadcast or multicast



Unicast

- One-to-one communication
- One sender and one receiver
- To send a datagram, the sender must know:
  - The IP address of the receiver
  - The port of the receiver

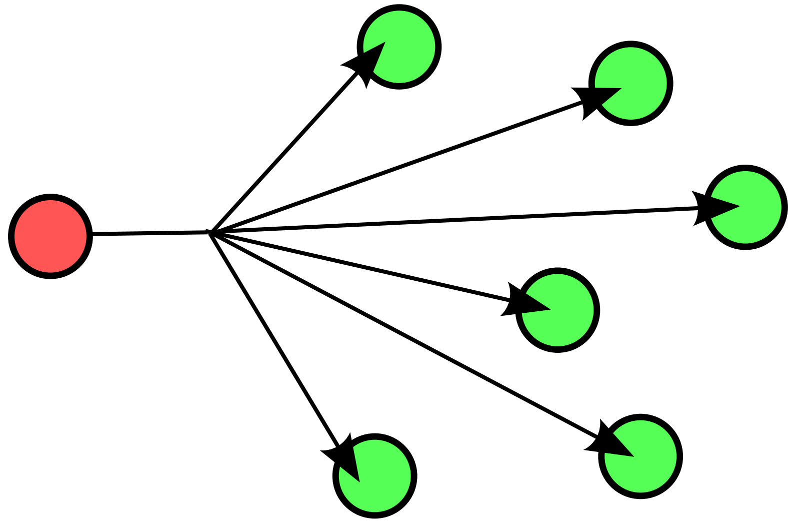
Think of it as a private conversation between two people



Broadcast

- One-to-all communication
- One sender and multiple receivers
- To send a datagram, the sender must know:
  - The subnet
  - The port
- `255.255.255.255` for all hosts

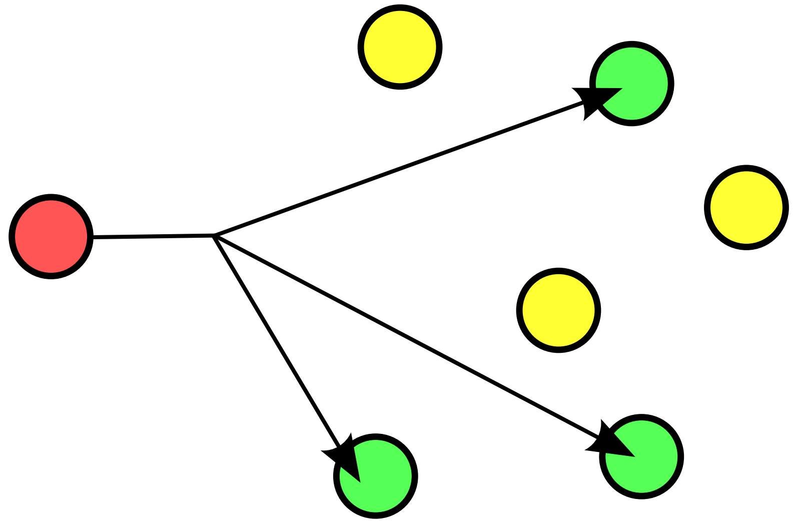
Think of it as a public announcement.



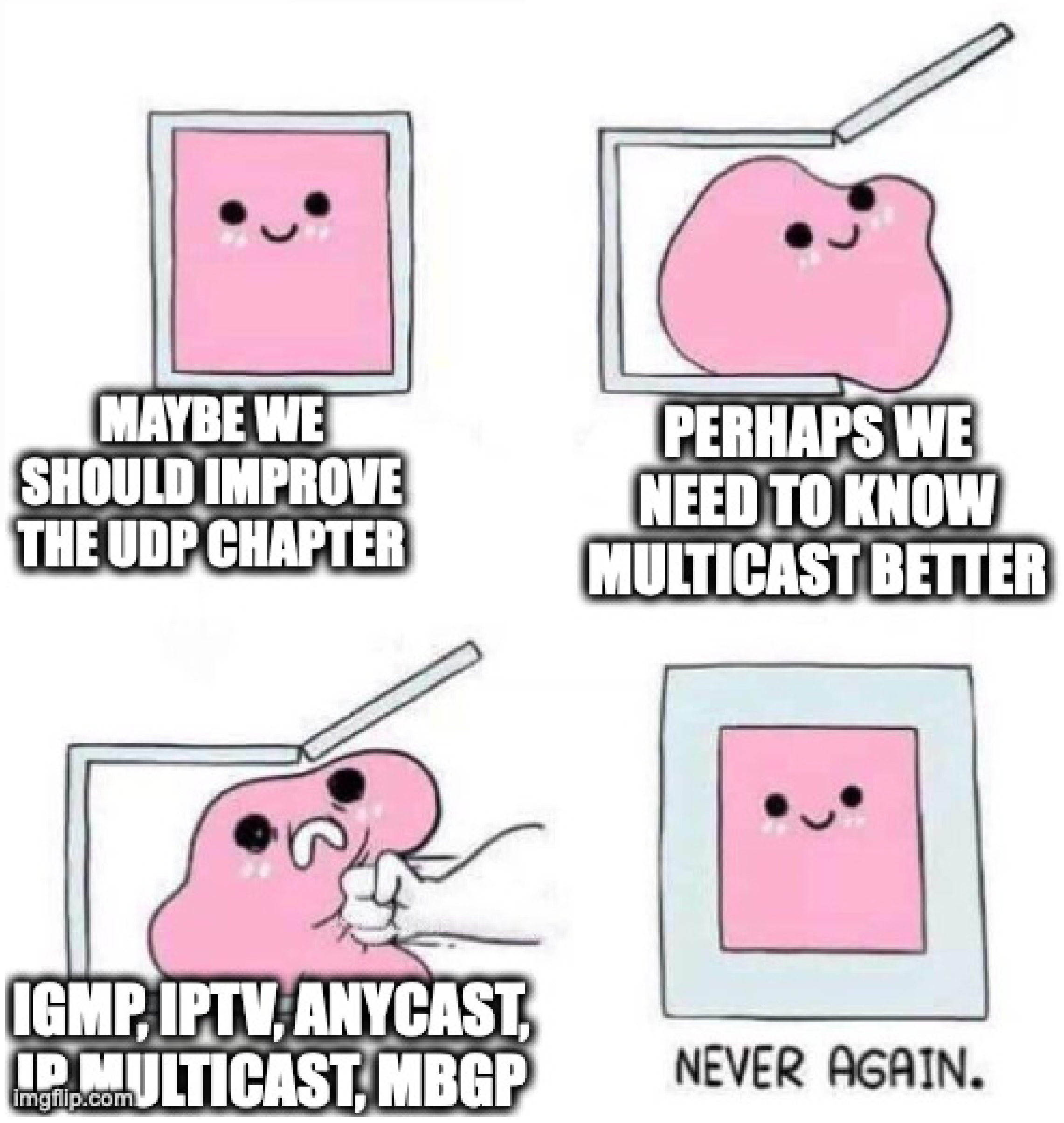
Multicast

- One-to-many communication
- One sender and some receivers
- To send a datagram, the sender must know:
  - The multicast address (between `239.0.0.0` and `239.255.255.255`)
  - The port

Think of it as a group conversation.



- Just as with broadcast, it can be blocked by routers
- Multicast is quite guaranteed **not** to work on the public Internet
- Made for the local network
- Multicast is a complex topic
- Not covered in depth in this course
- The course material contains some resources


MAYBE WE SHOULD IMPROVE THE UDP CHAPTER
PERHAPS WE NEED TO KNOW MULTICAST BETTER
IGMP, IPTV, ANYCAST, IP MULTICAST, MBGP
imgflip.com
NEVER AGAIN.

Messaging patterns

More details for this section in the course material. You can find other resources and alternatives as well.

Messaging patterns

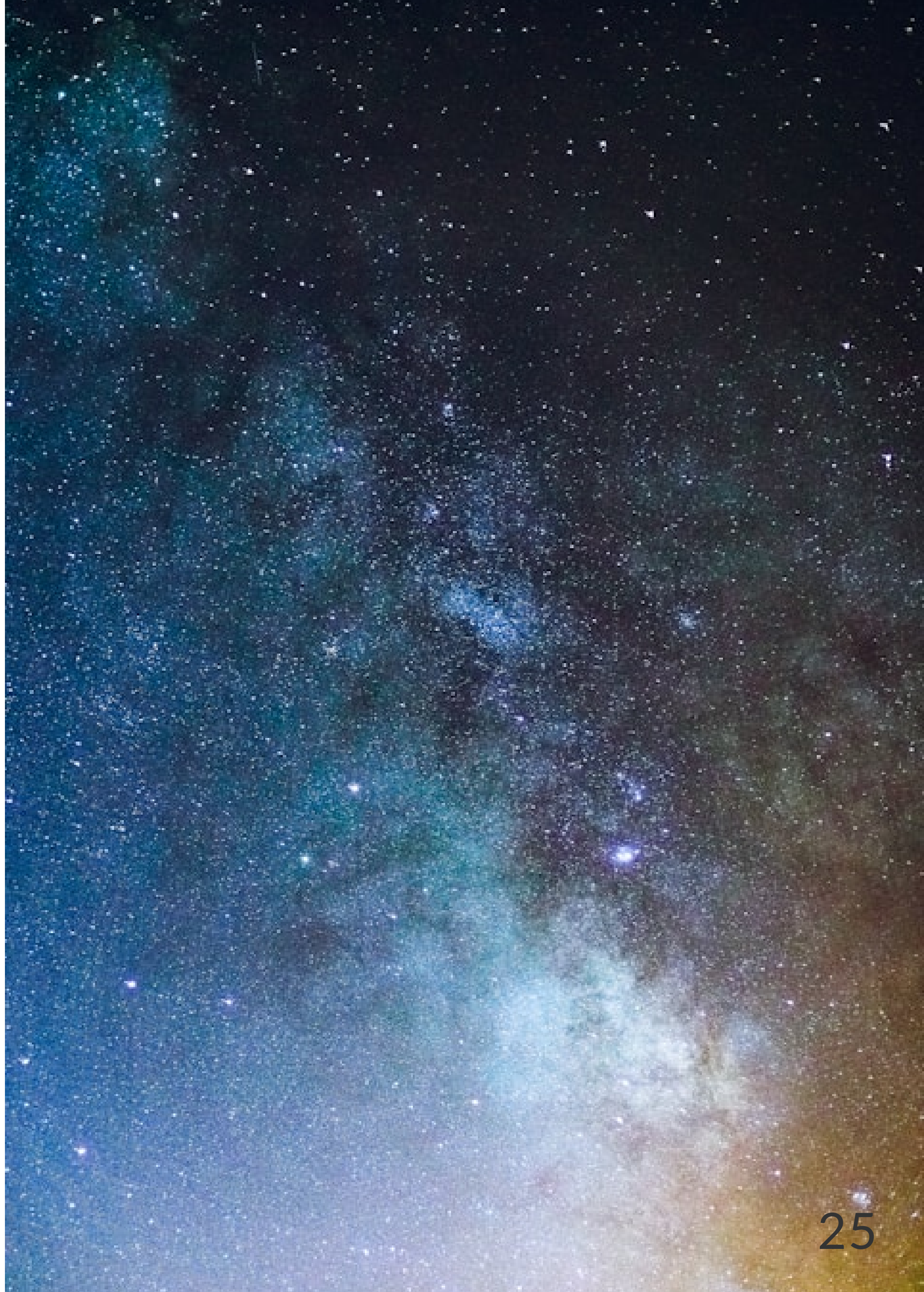
- Fire-and-forget
  - One-way communication
  - No response
  - No guarantee of delivery
- Request-response
  - Two-way communication
  - Response
  - Guarantee of delivery (manual)

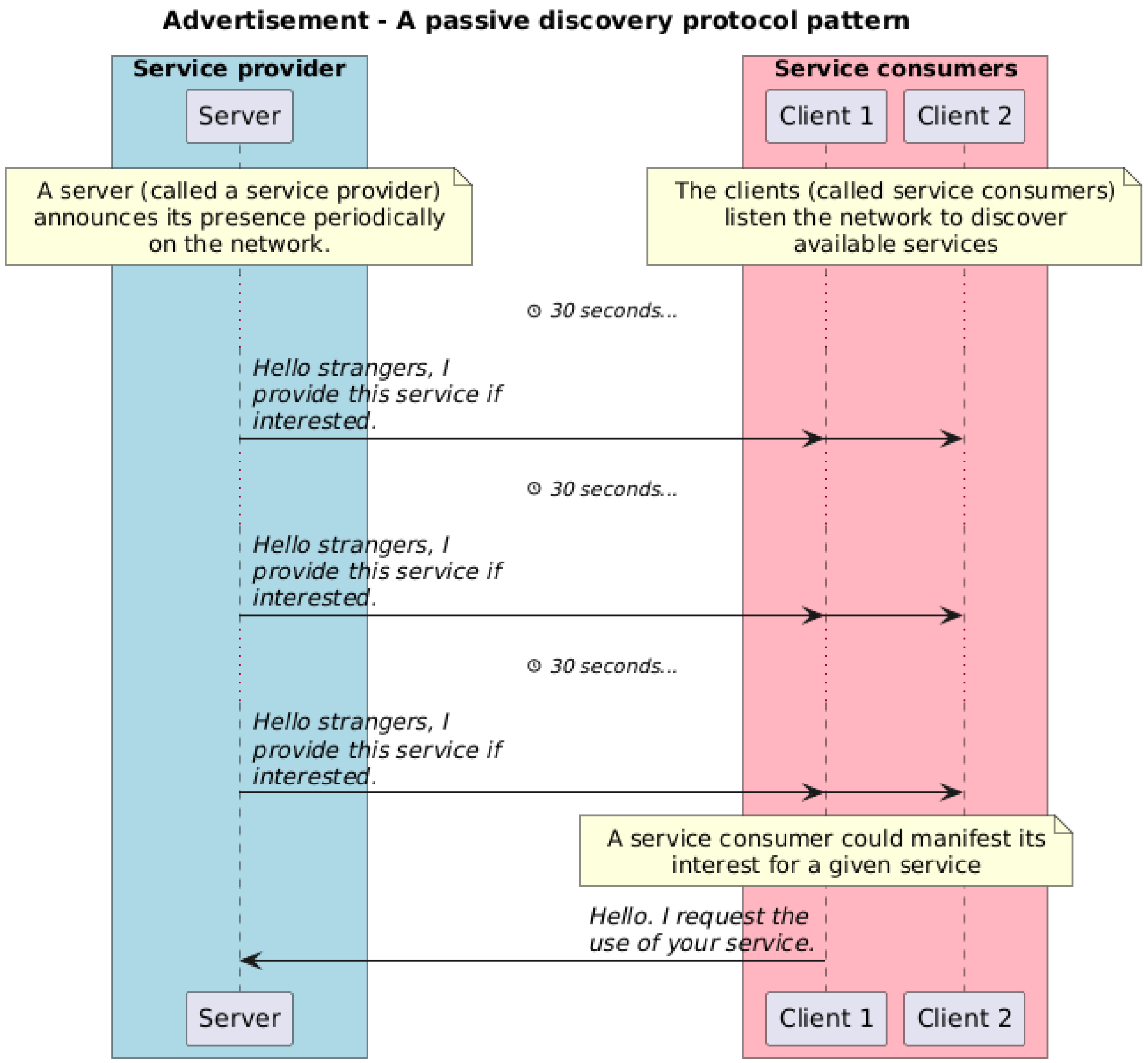
Service discovery protocols

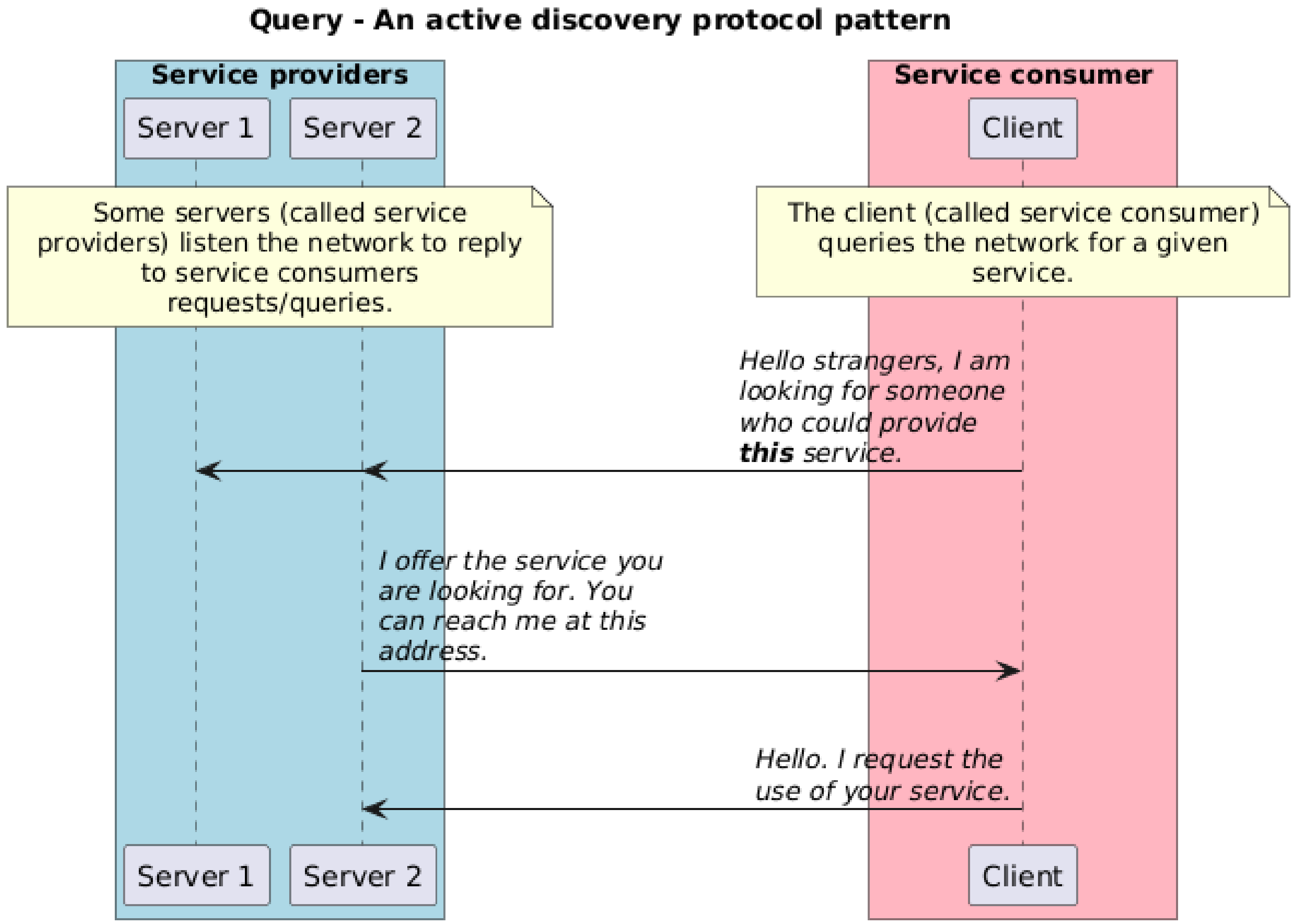
More details for this section in the course material. You can find other resources and alternatives as well.

Service discovery protocols

- Discover services on the network
- Two types of protocols
- Service discovery protocol patterns:
  - Advertisement (passive)
  - Query (active)




Advertisement - A passive discovery protocol pattern
Service provider
Server
Service consumers
Client 1
Client 2
A server (called a service provider) announces its presence periodically on the network.
The clients (called service consumers) listen the network to discover available services
30 seconds...
Hello strangers, I provide this service if interested.
30 seconds...
Hello strangers, I provide this service if interested.
30 seconds...
Hello strangers, I provide this service if interested.
A service consumer could manifest its interest for a given service
Hello. I request the use of your service.
Server
Client 1
Client 2


Query - An active discovery protocol pattern
Service providers
Server 1
Server 2
Service consumer
Client
Some servers (called service providers) listen the network to reply to service consumers requests/queries.
The client (called service consumer) queries the network for a given service.
*Hello strangers, I am looking for someone who could provide **this** service.*
I offer the service you are looking for. You can reach me at this address.
Hello. I request the use of your service.
Server 1
Server 2
Client

Questions

Do you have any questions?

Practical content

What will you do?

- Update your application protocol with the new knowledge you gained
- Learn to use the debugger
- Execute the code examples and run multiple emitters at the same time
- Explore the Java UDP programming template
- Implement the *"Temperature monitoring"* application (optional)

Find the practical content

You can find the practical content for this chapter on GitHub.

Finished? Was it easy? Was it hard?

Can you let us know what was easy and what was difficult for you during this chapter?

This will help us to improve the course and adapt the content to your needs. If we notice some difficulties, we will come back to you to help you.

➡️ GitHub Discussions

You can use reactions to express your opinion on a comment!

What will you do next?

- Learn the different ways to manage multiple clients at the same time with concurrency
- Update the two network applications to handle multiple clients (optional):
  - The *"Guess the number"* game
  - The *"Temperature monitoring"* application

Sources

- Main illustration by Possessed Photography on Unsplash
- Illustration by Aline de Nadai on Unsplash
- Illustration by Becky Phan on Unsplash
- Illustration by Jackson Simmer on Unsplash
- Illustration by Sam Dan Truong on Unsplash
- Illustration by Zuza Gałczyńska on Unsplash
- Illustration by Andy Holmes on Unsplash
- Illustration by Brent Olson on Unsplash