Course of

Robot Programming with ROS 2

Day 3

1. DDS Setup

ROS

Robot Operating System

ikerlan

Universidad Rey Juan Carlos

Intelligent Robotics Lab

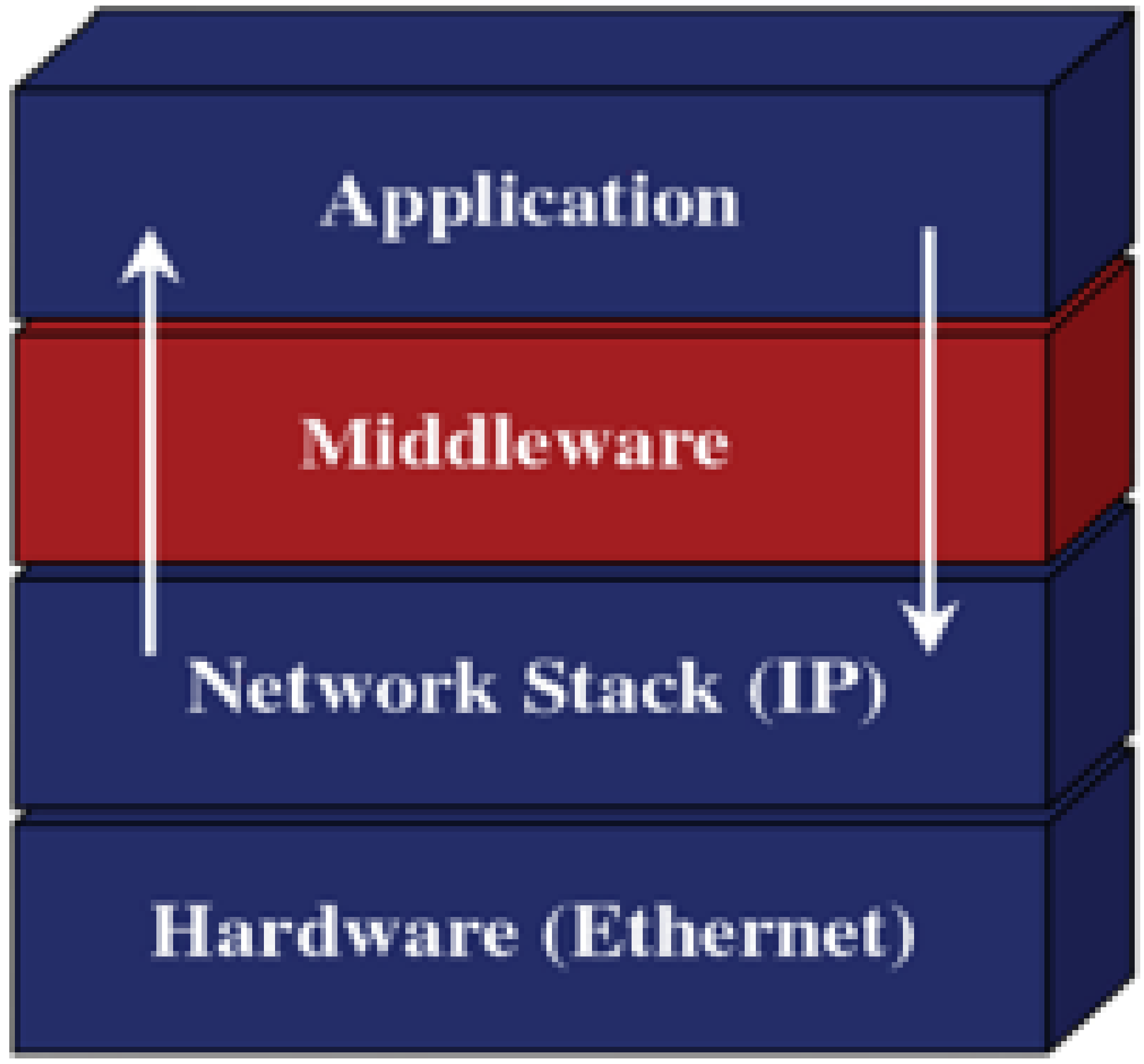
Introduction to DDS

Concept

- It is a network middleware between the application and the underlying operating system and network stack

- It is based on a publish-subscribe model.

- Uses Interface Description Language (IDL)



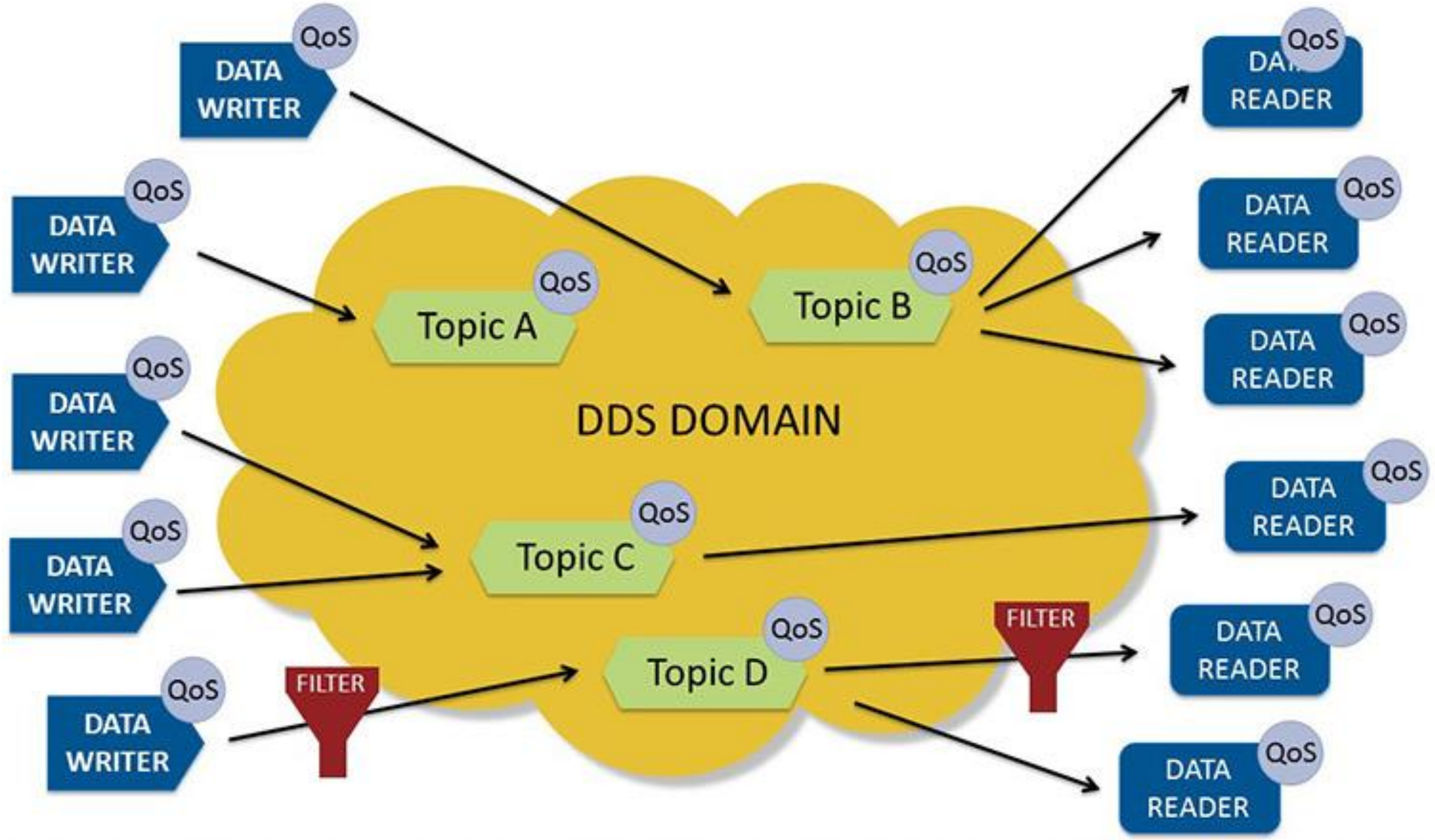
Introduction to DDS

features

- Customizable QoS, enabling efficient data transfer, guaranteed periodic samples and reliable delivery of samples.
- Multiple Communication Networks using domains.
- Symmetric architecture (decentralized nodes)
- Multiple transports like UDP, TCP, shared memory and the ability to define a new transport plug-in
- Supports unix systems, real time systems and windows.

Introduction to DDS

Overview



DDS setup

Profile configuration basics

- Majority of DDS implementations allows configurations files in xml format.
- A common way to load the XML is via environment variable:
  - CycloneDDS: export CYCLONEDDS_URI="path_to_file.xml"
  - *FastDDS*: export FASTRTPS_DEFAULT_PROFILES_FILE=path_to_file.xml
  - GurumDDS: export GURUMDDS_CONFIG=path_to_file.yaml

Intelligent Robotics Lab

DDS setup

Profile configuration basics

```
$ apt-get install ros-humble-rmw-fastrtps-cpp / apt-get install ros-humble-rmw-cyclonedds-cpp
```

```
$ export RMW_IMPLEMENTATION=rmw_cyclonedds_cpp / rmw_fastrtps_cp
```

```
$ ros2 doctor --report | grep middleware
```

DDS setup

Profile configuration basics

```
$ ros2 topic pub -r 10 /string_topic std_msgs/String "{data: \"Hello, ikerlan\"}" --qos-reliability best_effort

publisher: beginning loop

publishing #1: std_msgs.msg.String(data='Hello, ikerlan')

publishing #2: std_msgs.msg.String(data='Hello, ikerlan')

publishing #3: std_msgs.msg.String(data='Hello, ikerlan')
```

DDS setup

Profile configuration basics

```
$ ros2 topic info /string_topic --verbose
Type: std_msgs/msg/String


Publisher count: 1


Node name: _ros2cli_10994
Node namespace: /
Topic type: std_msgs/msg/String
Endpoint type: PUBLISHER
GID: 01.10.8d.d2.53.64.df.a1.59.d7.a8.e5.00.00.08.03.00.00.00.00.00.00.00.00
QoS profile:
  Reliability: BEST_EFFORT
  History (Depth): KEEP_LAST (1)
  Durability: TRANSIENT_LOCAL
  Lifespan: Infinite
  Deadline: Infinite
  Liveliness: AUTOMATIC
  Liveliness lease duration: Infinite


Subscription count: 0
```

DDS setup

Profile configuration basics

```
$ ros2 topic echo /string_topic --qos-reliability reliable
[WARN] [1685993218.136685772] [_ros2cli_11128]: New publisher discovered on topic '/string_topic', offering
incompatible QoS. No messages will be received from it. Last incompatible policy: RELIABILITY
```

```
$ ros2 topic echo /string_topic --qos best_effort
data: Hello, ikerlan
---
data: Hello, ikerlan
```

fastDDS setup

DDS setup

Sync and async publication configuration

- **Sync method:**
  - Wait until all the data has been sent.
  - Messages are added to a queue.
  - Publish() method is not finished until the data has been written into the transport mechanism (network socket or shared memory buffers)
- **Async method:**
  - Uses an internal thread to send the data.
  - Messages are store in a queue.
  - The async thread is woken up and notified new data has been added to the queue.
  - The publish() method finishes.
  - Data is sent in parallel execution by the async thread.

- Overall, snyc may be faster. However, it can block the user thread if there is a block call during the publish operation.

DDS setup

Sync and async publication configuration

```
<<?xml version="1.0" encoding="UTF-8" ?>
<profiles xmlns="http://www.eprosima.com/XMLSchemas/fastRTPS_Profiles">

    <!-- default publisher profile -->
    <publisher profile_name="default_publisher" is_default_profile="true">
        <historyMemoryPolicy>DYNAMIC</historyMemoryPolicy>
    </publisher>

    <!-- default subscriber profile -->
    <subscriber profile_name="default_subscriber" is_default_profile="true">
        <historyMemoryPolicy>DYNAMIC</historyMemoryPolicy>
    </subscriber>
    <!-- publisher profile for topic sync_topic -->
    <publisher profile_name="/string_topic">
        <historyMemoryPolicy>DYNAMIC</historyMemoryPolicy>
            <qos>
                <publishMode>
                    <kind>SYNCHRONOUS</kind>
                </publishMode>
            </qos>
    </publisher>
</profiles>
```

DDS setup

Sync and async publication configuration

```
export RMW_IMPLEMENTATION=rmw_fastrtps_cpp
export RMW_FASTRTPS_USE_QOS_FROM_XML=1
export FASTRTPS_DEFAULT_PROFILES_FILE=ws_path/src/dds_demos/config/fast/qos_config.xml
```

DDS setup

Sync and async publication configuration

```
$ ros2 topic pub -r 10 /string_topic std_msgs/String "{data: \"Hello, ikerlan sync\"}"
publisher: beginning loop

publishing #1: std_msgs.msg.String(data='Hello, ikerlan sync')

publishing #2: std_msgs.msg.String(data='Hello, ikerlan sync')

publishing #3: std_msgs.msg.String(data='Hello, ikerlan sync')
```

DDS setup

Resource limit configuration

- **Locators:**
  - Max unicast locators
  - Max multicast locators
- **Publishers/Subscribers:**
  - Initial number of matched subscribers
  - Maximum number of matched subscribers.
- **Buffers:**
  - Initial buffers number
  - Dynamic behavior (if true, new buffer will be created if there are not available)

DDS setup

Resource limit configuration

```
<<?xml version="1.0" encoding="UTF-8" ?>
<profiles xmlns="http://www.eprosima.com/XMLSchemas/fastRTPS_Profiles">

    <!-- default publisher profile -->
    <publisher profile_name="default_publisher" is_default_profile="true">
        <historyMemoryPolicy>DYNAMIC</historyMemoryPolicy>
    </publisher>

    <!-- default subscriber profile -->
    <subscriber profile_name="default_subscriber" is_default_profile="true">
        <historyMemoryPolicy>DYNAMIC</historyMemoryPolicy>
    </subscriber>
    <!-- publisher profile for topic sync_topic -->
    <publisher profile_name="/string_topic">
        <historyMemoryPolicy>DYNAMIC</historyMemoryPolicy>
            <qos>
                <publishMode>
                    <kind>SYNCHRONOUS</kind>
                </publishMode>
            </qos>
        <matchedSubscribersAllocation>
            <initial>0</initial>
            <maximum>1</maximum>
            <increment>0</increment>
        </matchedSubscribersAllocation>
    </publisher>
</profiles>
```

DDS setup

Resource limit configuration

```
export RMW_IMPLEMENTATION=rmw_fastrtps_cpp
export RMW_FASTRTPS_USE_QOS_FROM_XML=1
export FASTRTPS_DEFAULT_PROFILES_FILE=ws_path/src/dds_config/qos_config.xml
```

```
$ ros2 topic pub -r 10 /string_topic std_msgs/String "{data: \"Hello, ikerlan\"}"
publisher: beginning loop

publishing #1: std_msgs.msg.String(data='Hello, ikerlan')

publishing #2: std_msgs.msg.String(data='Hello, ikerlan')

publishing #3: std_msgs.msg.String(data='Hello, ikerlan')
```

DDS setup

Resource limit configuration

```
$ ros2 topic echo /string_topic
data: Hello, ikerlan
---
data: Hello, ikerlan
---
```

```
$ ros2 topic echo /string_topic
```

DDS setup

Scaling network traffic with domain ID

- Domains represents logical and isolated communication networks.
- Allows multiples applications running on the same set of hosts.
- Different domains will never exchange data.
- **Domain participants:**
  - It creates destroys and manages DDS objects.
  - An application participates in a domain by creating a domain participant for that domain id
  - Participants in the same domain are isolated from other participants.
  - A domain establishes a virtual network and links all participants that share the same domain id.
  - ROS 2 creates one participant for each process

DDS setup

Scaling network traffic with domain ID

```
$ ROS_DOMAIN_ID=5 ros2 topic pub -r 10 /string_topic std_msgs/String "{data: \"Hello, ikerlan from domain 5\"}"
publisher: beginning loop

publishing #1: std_msgs.msg.String(data='Hello, ikerlan from domain 5')
```

```
$ ROS_DOMAIN_ID=5 ros2 topic echo /string_topic
data: Hello, ikerlan
---
data: Hello, ikerlan
---
```

```
$ ros2 topic echo /string_topic
WARNING: topic [/string_topic] does not appear to be published yet
Could not determine the type for the passed topic
```

DDS setup

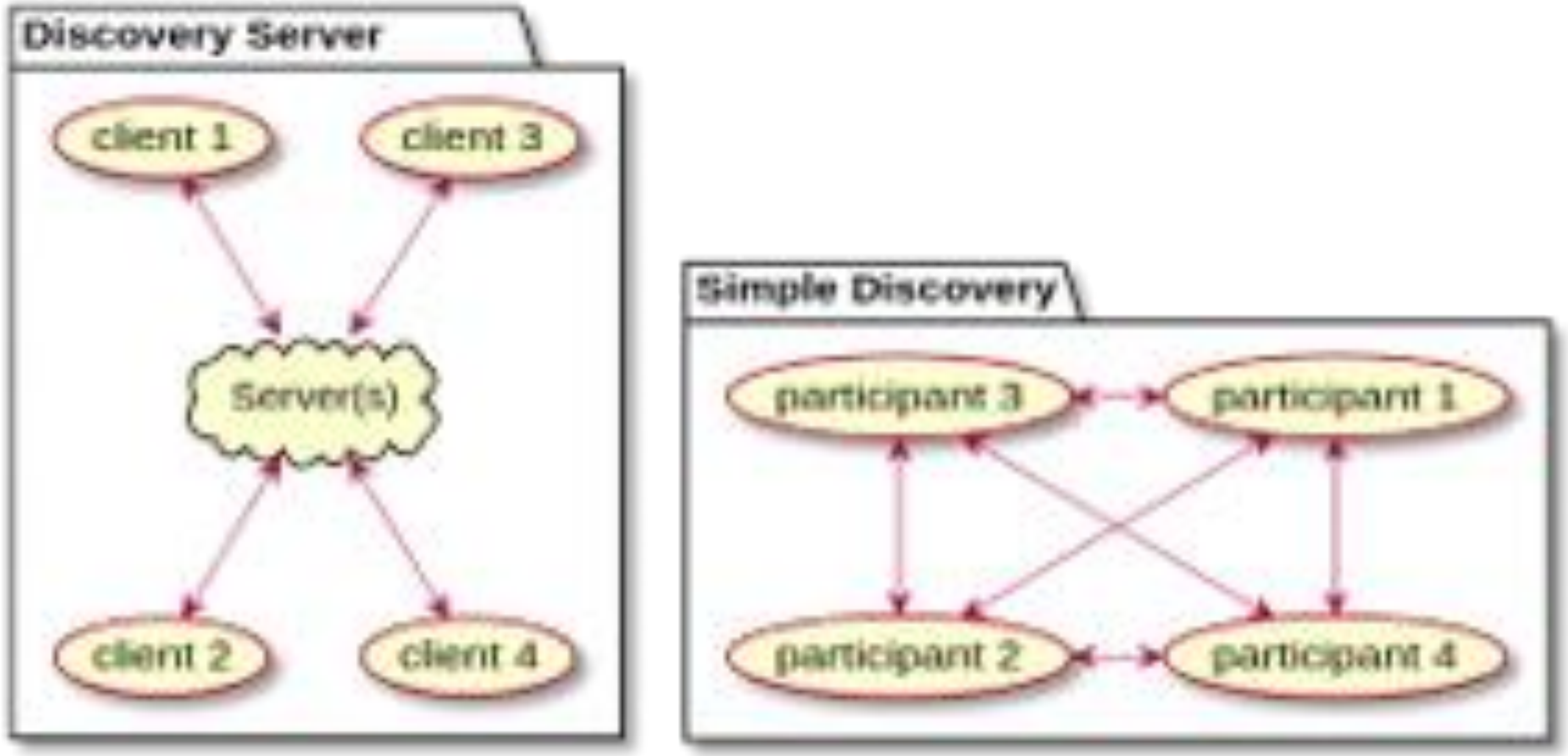
Scaling network traffic with domain ID

- Limited Domains IDs
- Considerable network usage, multicast data duplication
- No windowing mechanism, network congestion.

DDS setup

Scaling network traffic with fastDDS discovery server

- Based on standard DDS publishers and subscribers
- Ech ROS 2 node act as a client
- All clients share information with its servers.
- Servers use an identification to implement the communication.



DDS setup

Scaling network traffic with fastDDS discovery server

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<profiles xmlns="http://www.eprosima.com/XMLSchemas/fastRTPS_Profiles">
    <participant profile_name="server_example">
        <rtps>
            <prefix>44.53.00.5f.45.50.52.4f.53.49.4d.41</prefix>
            <builtin>
                <discovery_config>
                    <discoveryProtocol>SERVER</discoveryProtocol>
                    <discoveryServersList>
                        <RemoteServer prefix="44.53.01.5f.45.50.52.4f.53.49.4d.41">
                            <metatrafficUnicastLocatorList>
                                <locator>
                                    <udpv4>
                                        <address>127.0.0.1</address>
                                        <port>11812</port>
                                    </udpv4>
                                </locator>
                            </metatrafficUnicastLocatorList>
                        </RemoteServer>
                    </discoveryServersList>
                </discovery_config>

                <metatrafficUnicastLocatorList>
                    <locator>
                        <udpv4>
                            <address>127.0.0.1</address>
                            <port>11811</port>
                        </udpv4>
                    </locator>
                </metatrafficUnicastLocatorList>
            </builtin>
        </rtps>
    </participant>
    </profiles>
```

DDS setup

Scaling network traffic with fastDDS discovery server

```
$ fastdds discovery -i 0 -
x profilename@/path_ws/src/dds_demos/config/fast/server_config.xml
### Server is running ###
  Participant Type:   SERVER
  Server ID:          0
  Server GUID prefix: 44.53.00.5f.45.50.52.4f.53.49.4d.41
  Server Addresses:   UDPv4:[127.0.0.1]:11811
```

```
$ export
ROS_DISCOVERY_SERVER=127.0.0.1:11811
ros2 run demo_nodes_cpp listener
```

```
$ export
ROS_DISCOVERY_SERVER=127.0.0.1:11811
ros2 run demo_nodes_cpp talker
```

DDS setup

Managing large data rates

- Large data rates can result from sending large size data, a high message rate or a combination of both.

- Packages could be dropped because some transmitted amount of data fills the socket before it can be processed.

DDS setup

Managing large data rates

```xml
<profiles xmlns="http://www.eprosima.com/XMLSchemas/fastRTPS_Profiles">
<participant profile_name="participant_profile_ros2_large_files" is_default_profile="true">
    <rtps>
        <name>large_files_profile</name>
        <sendSocketBufferSize>10194304</sendSocketBufferSize>
        <listenSocketBufferSize>10194304</listenSocketBufferSize>
    </rtps>
</participant>
</profiles>
```

DDS setup

Managing large data rates

```
$ ros2 run dds_demos large_file
[INFO] [1686058666.904160367] [large_file]: Publishing image
[INFO] [1686058667.055932315] [large_file]: Publishing image
```

```
$ ros2 topic hz /output_image
average rate: 2.003
    min: 0.247s max: 0.749s std dev: 0.20502s window: 3
average rate: 2.171
    min: 0.247s max: 0.749s std dev: 0.17218s window: 6
average rate: 2.248
    min: 0.247s max: 0.749s std dev: 0.15692s window: 9
```

DDS setup

Managing large data rates

```
$ sudo sysctl -a | grep net.core | grep wmem
net.core.wmem_default = 212992
net.core.wmem_max = 212992
```

```
$ sudo sysctl -a | grep net.core | grep rmem
net.core.rmem_default = 212992
net.core.rmem_max = 212992
```

DDS setup

Managing large data rates

```
$ sudo sysctl -w net.core.wmem_max=10194304
net.core.wmem_max = 10194304
```

```
$ sudo sysctl -w net.core.rmem_max=10194304
net.core.rmem_max = 10194304
```

DDS setup

Managing large data rates

```
export FASTRTPS_DEFAULT_PROFILES_FILE=/path_ws/src/ikerlan/dds_demos/config/fast/large_data_config.xml
```

```
ros2 topic hz /output_image
average rate: 3.998
    min: 0.244s max: 0.256s std dev: 0.00436s window: 6
average rate: 4.003
    min: 0.244s max: 0.257s std dev: 0.00435s window: 11
average rate: 4.000
    min: 0.244s max: 0.257s std dev: 0.00396s window: 15
```

cycloneDDS setup

DDS setup

Resource limit configuration

- Wait-For-Historical-Data Completion (WHC):
  - Reader may request historical data to catch up with the previously published information when joining a topic
  - Allows the writer to pause or suspend publishing new data until the requested historical data is fully delivered to the reader.

- High-Water Mark:
  - Is a threshold to control the amount of data that can accumulate before a writer is suspended.

Intelligent Robotics Lab

DDS setup

Resource limit configuration

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<CycloneDDS xmlns="https://cdds.io/config"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://cdds.io/config https://raw.githubusercontent.com/eclipse-
cyclonedds/cyclonedds/master/etc/cyclonedds.xsd">
    <Domain Id="any">
        <Internal>
            <Watermarks>
                <WhcHigh>100kB</WhcHigh>
            </Watermarks>
        </Internal>
    </Domain>
</CycloneDDS>
```

DDS setup

Scaling network traffic, with unicast

- Reduces the middleware setup time.
- limits the connections to those strictly necessary.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<CycloneDDS xmlns="https://cdds.io/config" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://cdds.io/config https://raw.githubusercontent.com/eclipse-cyclonedds/cyclonedds/master/etc/cyclonedds.xsd">
    <Domain Id="any">
        <General>
            <AllowMulticast>false</AllowMulticast>
            <EnableMulticastLoopback>false</EnableMulticastLoopback>
        </General>
        <Discovery>
            <ParticipantIndex>auto</ParticipantIndex>
            <Peers>
                <Peer Address="localhost"/>
            </Peers>
        </Discovery>
    </Domain>
</CycloneDDS>
```

DDS setup

Working with large data rates

```
<?xml version="1.0" encoding="UTF-8" ?>
<CycloneDDS xmlns="https://cdds.io/config" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="https://cdds.io/config https://raw.githubusercontent.com/eclipse-
cyclonedds/cyclonedds/master/etc/cyclonedds.xsd">
    <Domain Id="any">
    <General>
        <MaxMessageSize>10194304B</MaxMessageSize>
    </General>
    </Domain>
</CycloneDDS>
```

Future of DDS configuration using ROS2

- ROS 2 Iron Irwini changes:
  - ROS_AUTOMATIC_DISCOVERY_RANGE
    - SUBNET: Same as humble.
    - LOCALHOST: Discover nodes only on the local machine.
    - OFF: No attempt to discover any node.
    - SYSTEM_DEFAULT: It wont change DDSs configuration, useful when xml is provided.
  - ROS_STATIC_PEERS: Unicast addresses.

Resources

- https://fast-dds.docs.eprosima.com/en/latest/index.html
- https://community.rti.com/rti-doc/45d/ndds.4.5d/doc/pdf/RTI_DDS_UsersManual.pdf
- https://cyclonedds.io/docs/cyclonedds/latest/index.html