13. ROS2 Distributed Communication

1. Concept

Multi-machine communication, or distributed communication, refers to a communication strategy that enables data exchange between different hosts over a network.

ROS2 itself is a distributed communication framework that facilitates communication between different devices. The middleware underlying ROS2 is DDS. When running on the same network, distributed communication is achieved through the DDS domain ID mechanism (ROS_DOMAIN_ID). The general process is as follows: Before starting a node, you can set the domain ID value. Nodes with the same domain ID can freely discover and communicate with each other. Conversely, if the domain ID values are different, this communication is impossible. By default, all nodes start with domain ID 0. In other words, as long as they are on the same network, distributed communication can be achieved between different nodes on different ROS2 devices without any configuration.

Distributed communication has a wide range of application scenarios, including unmanned vehicle platooning, drone platooning, and remote control. These data exchanges all rely on distributed communication.

2. Implementation

2.1. Default Implementation

Distributed communication is achieved by simply placing the master and slave devices (you can have multiple devices) on the same network. For example, the master and slave devices can be connected to the same WiFi or router.

In Windows, setting the virtual machine to "bridge mode" will place them on the same network as the host.

Test:

Here, we assume we have two hosts, A and B. These can be any network-capable host, such as a virtual machine, Raspberry Pi, Jetson, x86/ARM host, or card motherboard. They only need to have the same version of the ROS2 environment installed.

1. Execute on host A:

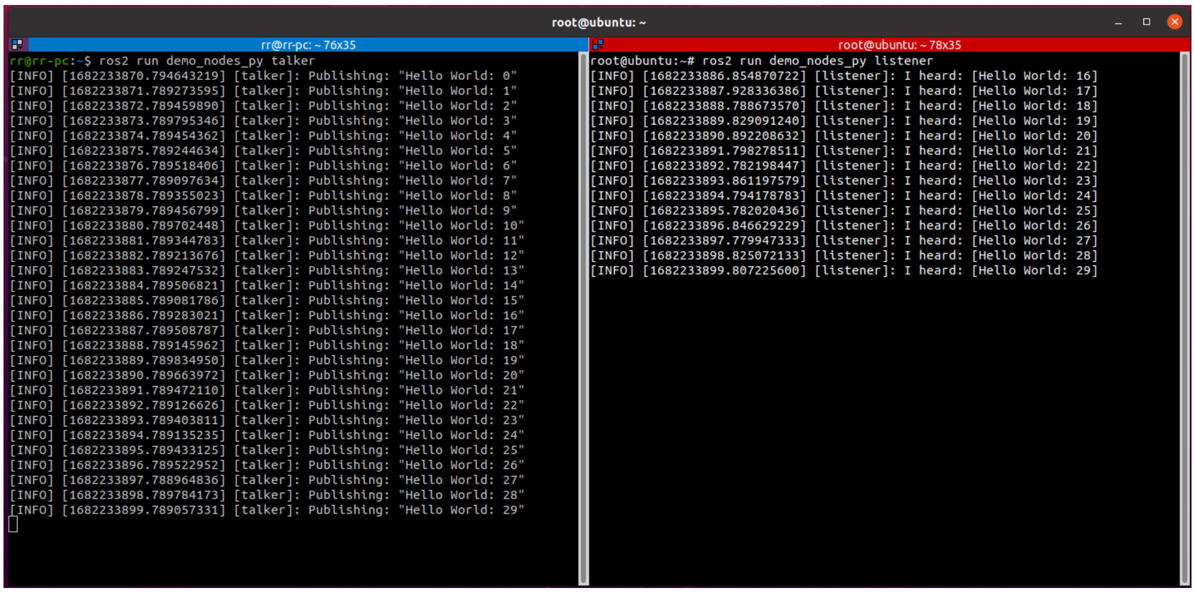
This demonstration shows the car running in Docker. Docker uses host mode. Simply put, host mode means the car shares the same network, so executing the command is identical to running the command on the car.

```
ros2 run demo_nodes_py talker
```

2. Host B executes:

```
ros2 run demo_nodes_py listener
```

If the following display appears: The slaves can subscribe to the topics published by the host, multi-machine communication has been achieved.



```
rr@rr-pc:~$ ros2 run demo_nodes_py talker
[INFO] [1682233870.794643219] [talker]: Publishing: "Hello World: 0"
[INFO] [1682233871.789273595] [talker]: Publishing: "Hello World: 1"
[INFO] [1682233872.789459890] [talker]: Publishing: "Hello World: 2"
[INFO] [1682233873.789795346] [talker]: Publishing: "Hello World: 3"
[INFO] [1682233874.789454362] [talker]: Publishing: "Hello World: 4"
[INFO] [1682233875.789244634] [talker]: Publishing: "Hello World: 5"
[INFO] [1682233876.789518406] [talker]: Publishing: "Hello World: 6"
[INFO] [1682233877.789097634] [talker]: Publishing: "Hello World: 7"
[INFO] [1682233878.789355023] [talker]: Publishing: "Hello World: 8"
[INFO] [1682233879.789456799] [talker]: Publishing: "Hello World: 9"
[INFO] [1682233880.789702448] [talker]: Publishing: "Hello World: 10"
[INFO] [1682233881.789344783] [talker]: Publishing: "Hello World: 11"
[INFO] [1682233882.789213676] [talker]: Publishing: "Hello World: 12"
[INFO] [1682233883.789247532] [talker]: Publishing: "Hello World: 13"
[INFO] [1682233884.789506821] [talker]: Publishing: "Hello World: 14"
[INFO] [1682233885.789081786] [talker]: Publishing: "Hello World: 15"
[INFO] [1682233886.789283021] [talker]: Publishing: "Hello World: 16"
[INFO] [1682233887.789508787] [talker]: Publishing: "Hello World: 17"
[INFO] [1682233888.789145962] [talker]: Publishing: "Hello World: 18"
[INFO] [1682233889.789834950] [talker]: Publishing: "Hello World: 19"
[INFO] [1682233890.789663972] [talker]: Publishing: "Hello World: 20"
[INFO] [1682233891.789472110] [talker]: Publishing: "Hello World: 21"
[INFO] [1682233892.789126626] [talker]: Publishing: "Hello World: 22"
[INFO] [1682233893.789403811] [talker]: Publishing: "Hello World: 23"
[INFO] [1682233894.789135235] [talker]: Publishing: "Hello World: 24"
[INFO] [1682233895.789433125] [talker]: Publishing: "Hello World: 25"
[INFO] [1682233896.789522952] [talker]: Publishing: "Hello World: 26"
[INFO] [1682233897.788964836] [talker]: Publishing: "Hello World: 27"
[INFO] [1682233898.789784173] [talker]: Publishing: "Hello World: 28"
[INFO] [1682233899.789057331] [talker]: Publishing: "Hello World: 29"
```

```
root@ubuntu:~# ros2 run demo_nodes_py listener
[INFO] [1682233886.854870722] [listener]: I heard: [Hello World: 16]
[INFO] [1682233887.928336386] [listener]: I heard: [Hello World: 17]
[INFO] [1682233888.788673570] [listener]: I heard: [Hello World: 18]
[INFO] [1682233889.829091240] [listener]: I heard: [Hello World: 19]
[INFO] [1682233890.892208632] [listener]: I heard: [Hello World: 20]
[INFO] [1682233891.798278511] [listener]: I heard: [Hello World: 21]
[INFO] [1682233892.782198447] [listener]: I heard: [Hello World: 22]
[INFO] [1682233893.861197579] [listener]: I heard: [Hello World: 23]
[INFO] [1682233894.794178783] [listener]: I heard: [Hello World: 24]
[INFO] [1682233895.782020436] [listener]: I heard: [Hello World: 25]
[INFO] [1682233896.846629229] [listener]: I heard: [Hello World: 26]
[INFO] [1682233897.779947333] [listener]: I heard: [Hello World: 27]
[INFO] [1682233898.825072133] [listener]: I heard: [Hello World: 28]
[INFO] [1682233899.807225600] [listener]: I heard: [Hello World: 29]
```

2.2. Distributed Network Grouping

If you are currently using other robots in your network, you can also set up a group for your robot to prevent interference from other robots.

ROS2 provides a DOMAIN mechanism, similar to grouping. Only computers in the same DOMAIN can communicate. We can assign both the host (car) and the slave (virtual machine) to the same group by adding the following line to their .bashrc files:

```
$ export ROS_DOMAIN_ID=<your_domain_id>
```

If the host (car) and the slave (virtual machine) are assigned different IDs, they will not be able to communicate, thus achieving the purpose of grouping.

2.2.1 Example 1

1. Execute on the host (car):

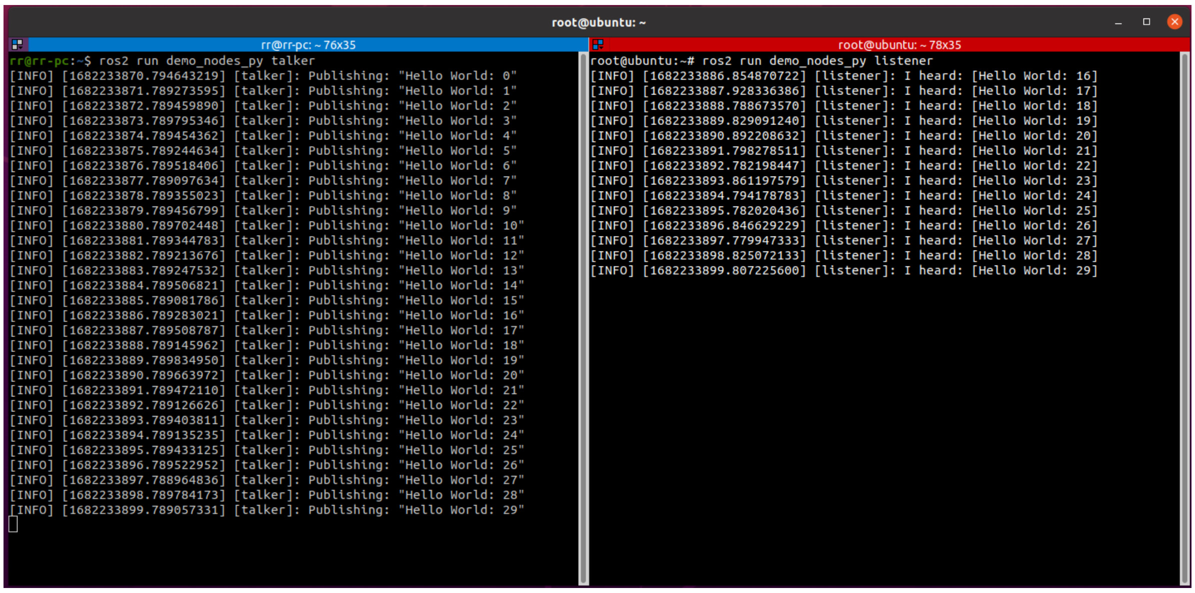
This demonstrates the car running in Docker, which uses host network mode. Simply put, host mode means the car shares the same network with the car, so the execution is identical to running on the car.

```
echo "export ROS_DOMAIN_ID=6" >> ~/.bashrc # The value 6 here refers to the ROS_DOMAIN_ID. It doesn't have to be 6, as long as it conforms to the ROS_DOMAIN_ID rules.
source ~/.bashrc
ros2 run demo_nodes_py talker
```

2. Simultaneously, execute the following command on the slave machine [virtual machine]:

```
echo "export ROS_DOMAIN_ID=6" >> ~/.bashrc # This value matches the value on the master machine.
source ~/.bashrc
ros2 run demo_nodes_py listener
```

If the following message appears: The slave machine can subscribe to the topic published by the master machine in a timely manner, indicating that grouped multi-machine communication has been achieved.



```
rr@rr-pc:~$ ros2 run demo_nodes_py talker
[INFO] [1682233870.794643219] [talker]: Publishing: "Hello World: 0"
[INFO] [1682233871.789273595] [talker]: Publishing: "Hello World: 1"
[INFO] [1682233872.789459890] [talker]: Publishing: "Hello World: 2"
[INFO] [1682233873.789795346] [talker]: Publishing: "Hello World: 3"
[INFO] [1682233874.789454362] [talker]: Publishing: "Hello World: 4"
[INFO] [1682233875.789244634] [talker]: Publishing: "Hello World: 5"
[INFO] [1682233876.789518406] [talker]: Publishing: "Hello World: 6"
[INFO] [1682233877.789097634] [talker]: Publishing: "Hello World: 7"
[INFO] [1682233878.789355023] [talker]: Publishing: "Hello World: 8"
[INFO] [1682233879.789456799] [talker]: Publishing: "Hello World: 9"
[INFO] [1682233880.789702448] [talker]: Publishing: "Hello World: 10"
[INFO] [1682233881.789344783] [talker]: Publishing: "Hello World: 11"
[INFO] [1682233882.789213676] [talker]: Publishing: "Hello World: 12"
[INFO] [1682233883.789247532] [talker]: Publishing: "Hello World: 13"
[INFO] [1682233884.789506821] [talker]: Publishing: "Hello World: 14"
[INFO] [1682233885.789081786] [talker]: Publishing: "Hello World: 15"
[INFO] [1682233886.789283021] [talker]: Publishing: "Hello World: 16"
[INFO] [1682233887.789508787] [talker]: Publishing: "Hello World: 17"
[INFO] [1682233888.789145962] [talker]: Publishing: "Hello World: 18"
[INFO] [1682233889.789834950] [talker]: Publishing: "Hello World: 19"
[INFO] [1682233890.789663972] [talker]: Publishing: "Hello World: 20"
[INFO] [1682233891.789472110] [talker]: Publishing: "Hello World: 21"
[INFO] [1682233892.789126626] [talker]: Publishing: "Hello World: 22"
[INFO] [1682233893.789403811] [talker]: Publishing: "Hello World: 23"
[INFO] [1682233894.789135235] [talker]: Publishing: "Hello World: 24"
[INFO] [1682233895.789433125] [talker]: Publishing: "Hello World: 25"
[INFO] [1682233896.789522952] [talker]: Publishing: "Hello World: 26"
[INFO] [1682233897.788964836] [talker]: Publishing: "Hello World: 27"
[INFO] [1682233898.789784173] [talker]: Publishing: "Hello World: 28"
[INFO] [1682233899.789057331] [talker]: Publishing: "Hello World: 29"
```

```
root@ubuntu:~# ros2 run demo_nodes_py listener
[INFO] [1682233886.854870722] [listener]: I heard: [Hello World: 16]
[INFO] [1682233887.928336386] [listener]: I heard: [Hello World: 17]
[INFO] [1682233888.788673570] [listener]: I heard: [Hello World: 18]
[INFO] [1682233889.829091240] [listener]: I heard: [Hello World: 19]
[INFO] [1682233890.892208632] [listener]: I heard: [Hello World: 20]
[INFO] [1682233891.798278511] [listener]: I heard: [Hello World: 21]
[INFO] [1682233892.782198447] [listener]: I heard: [Hello World: 22]
[INFO] [1682233893.861197579] [listener]: I heard: [Hello World: 23]
[INFO] [1682233894.794178783] [listener]: I heard: [Hello World: 24]
[INFO] [1682233895.782020436] [listener]: I heard: [Hello World: 25]
[INFO] [1682233896.846629229] [listener]: I heard: [Hello World: 26]
[INFO] [1682233897.779947333] [listener]: I heard: [Hello World: 27]
[INFO] [1682233898.825072133] [listener]: I heard: [Hello World: 28]
[INFO] [1682233899.807225600] [listener]: I heard: [Hello World: 29]
```

2.2.2 Case 2

Controlling turtle movement through distributed communication

- Host A run command

```
ros2 run turtlesim turtlesim_node
```

- Host B run command

```
ros2 run turtlesim turtle_teleop_key
```

3. Notes

Setting the ROS_DOMAIN_ID value is not arbitrary and is subject to certain constraints:

1. It is recommended that the ROS_DOMAIN_ID value be between [0, 101], inclusive;
2. The total number of nodes within each domain ID is limited and must be less than or equal to 120;
3. If the domain ID is 101, the total number of nodes in that domain must be less than or equal to 54.

4. DDS Domain ID Calculation Rules (Advanced Knowledge)

The calculation rules for domain ID values are as follows:

1. DDS is based on the TCP/IP or UDP/IP network communication protocol. Network communication requires a specified port number, which is represented by a 2-byte unsigned integer and has a value range of [0, 65535].
2. Port number assignment is also based on specific rules and cannot be used arbitrarily. According to the DDS protocol, port 7400 is used as the starting port, meaning the available ports are [7400, 65535]. Furthermore, according to the DDS protocol, each domain ID occupies 250 ports by default. Therefore, the number of domain IDs is: (65535 - 7400) / 250 = 232, corresponding to a value range of [0, 231].
3. The operating system also sets some reserved ports. When using ports in DDS, you need to avoid these reserved ports to avoid conflicts. Different operating systems have different reserved ports. As a result, in Linux, the available domain IDs are [0, 101] and [215-231]. In

Windows and Mac, the available domain IDs are [0, 166]. In summary, for compatibility across multiple platforms, it is recommended that domain IDs be within the range of [0, 101].

4. Each domain ID occupies 250 ports by default, and each ROS2 node requires two ports. Furthermore, according to the DDS protocol, ports 1 and 2 within each domain ID's port range are the Discovery Multicast port and the User Multicast port. Ports 11 and 12 onwards are the Discovery Unicast port and the User Unicast port for the first node in the domain, and subsequent nodes occupy these ports in descending order. Therefore, the maximum number of nodes in a domain ID is: (250 - 10) / 2 = 120.
5. Special case: For domain ID 101, the second half of the ports are reserved by the operating system, and the maximum number of nodes is 54.

The above calculation rules are sufficient for understanding.