Multi-machine communication configuration

4.1, Concept

Multi-machine communication, or distributed communication, refers to a communication strategy that can achieve data interaction between different hosts through the network.

ROS2 itself is a distributed communication framework that can easily achieve communication between different devices. The middleware based on ROS2 is DDS. When in the same network, distributed communication can be achieved through the domain ID mechanism (ROS_DOMAIN_ID) of DDS. The general process is: before starting the node, you can set the value of the domain ID. If the domain ID of different nodes is the same, they can be freely discovered and communicated. Conversely, if the domain ID values are different, it cannot be achieved. By default, the domain ID used when all nodes are started is 0. In other words, as long as you ensure that you are in the same network, you do not need to do any configuration, and different nodes on different ROS2 devices can achieve distributed communication.

The application scenarios of distributed communication are relatively wide, such as unmanned vehicle formations, drone formations, remote control, etc. The interaction of these data depends on distributed communication.

4.2, Implementation

4.2.1, Default Implementation

Distributed communication is achieved by simply placing the host and slaves [can have multiple] in the same network. For example, the host and slaves are connected to the same WiFi or the same router.

In Windows, the virtual machine is set to [bridge mode] to be in the same network as the host.

Test:

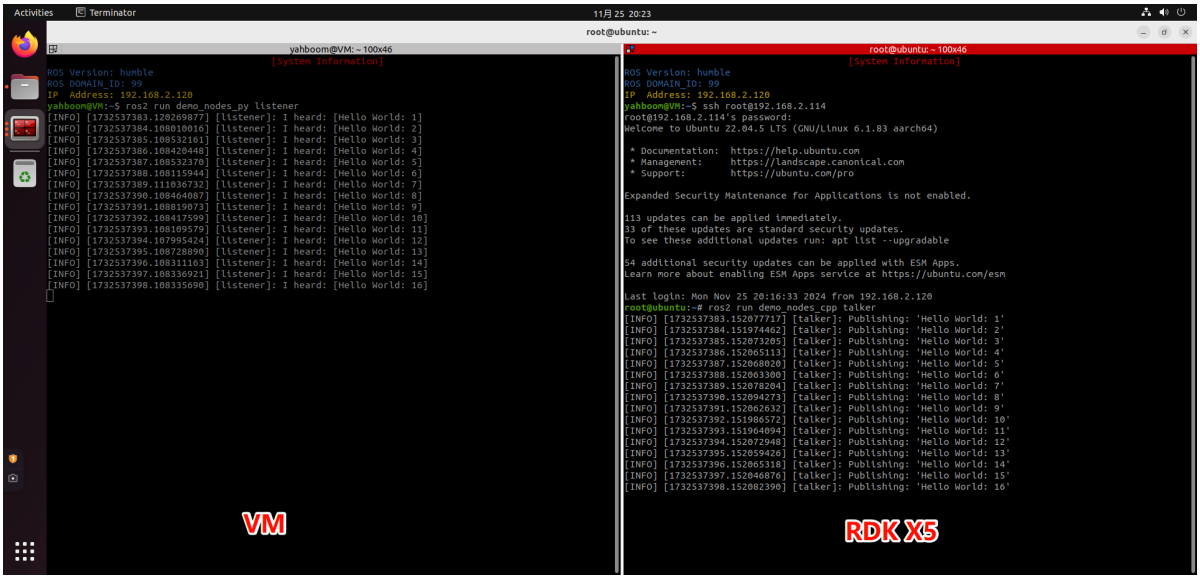
1. Execute on the host side [car]:

```
ros2 run demo_nodes_cpp talker
```

2. Execute on the slave side [virtual machine] at the same time:

```
ros2 run demo_nodes_py listener
```

If the following is displayed: The topic published by the host side can be subscribed to by the slave side in time, it means that multi-machine communication has been achieved



4.2.2, Distributed network grouping

Assuming that there are other robots in use in the network you are currently in, in order to avoid interference from other robots, you can also set a group for your robot.

ROS2 provides a DOMAIN mechanism, just like grouping, only computers in the same DOMAIN can communicate. We can add such a configuration in ~/.bashrc of the host [car] and the slave [virtual machine] to assign them to a group:

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

If the IDs assigned to the host [car] and the slave [virtual machine] are different, the two cannot communicate and achieve the purpose of grouping.

Note: The factory image of the RDK X5 Robot car has been configured with multi-machine communication with the PC virtual machine, and ROS_DOMAIN_ID=99. No need to perform the following operations, you can directly refer to 4.2.1 for testing.

1. Execute on the host side [car]:

```
echo "export ROS_DOMAIN_ID=99" >> ~/.bashrc # 99 is not necessary, as long as it complies with the rules of ROS_DOMAIN_ID
source ~/.bashrc
ros2 run demo_nodes_py talker
```

2. Execute on the slave side [virtual machine] at the same time:

```
echo "export ROS_DOMAIN_ID=66" >> ~/.bashrc # The value here is consistent with the value on the host side
source ~/.bashrc
ros2 run demo_nodes_py listener
```

The slave side can subscribe to the topic published by the host side in time, indicating that grouped multi-machine communication has been realized

> [Note] When setting the value of ROS_DOMAIN_ID, it is not arbitrary, but also has certain constraints:
>
> 1. It is recommended that the value of ROS_DOMAIN_ID be between [0,101], including 0 and 101;
> 2. The total number of nodes in each domain ID is limited and needs to be less than or equal to 120;
> 3. If the domain ID is 101, then the total number of nodes in the domain needs to be less than or equal to 54.

4.3. Calculation rules for DDS domain ID values

The relevant calculation rules for domain ID values are as follows:

1. DDS is based on TCP/IP or UDP/IP network communication protocols. When communicating on the network, a port number needs to be specified. The port number is represented by a 2-byte unsigned integer and its value range is between [0,65535];
2. The allocation of port numbers also has its rules and cannot be used arbitrarily. According to the DDS protocol, 7400 is used as the starting port, that is, the available ports are [7400,65535]. It is also known that according to the default DDS protocol, each domain ID occupies 250 ports, so the number of domain IDs is: (65535-7400)/250 = 232 (ports), and the corresponding value range is [0,231];
3. The operating system will also set some reserved ports. When using ports in DDS, you need to avoid these reserved ports to avoid conflicts during use. Different operating systems have different reserved ports. The final result is that under Linux, the available domain IDs are [0,101] and [215-231], and the available domain IDs in Windows and Mac are [0,166]. In summary, in order to be compatible with multiple platforms, it is recommended that the domain ID be in the range of [0,101].
4. Each domain ID occupies 250 ports by default, and each ROS2 node needs to occupy two ports. In addition, according to the DDS protocol, in the port segment of each domain ID, the 1st and 2nd ports are the Discovery Multicast port and the User Multicast port. Starting from the 11th and 12th ports, they are the Discovery Unicast port and the User Unicast port of the first node in the domain. The ports occupied by subsequent nodes are extended in sequence. Then the maximum number of nodes in a domain ID is: (250-10)/2 = 120 (nodes);
5. Special case: When the domain ID value is 101, the second half of the ports belong to the reserved ports of the operating system, and the maximum number of nodes is 54.

The above calculation rules can be understood.