

자율주행 데브코스

ROS 노드 통신 프로그래밍

(주)자이트론

허성민

smher@xytron.co.kr



Contents

노드 통신을 위한 패키지 만들기

1:N, N:1, N:N 통신

나만의 메시지 만들기

다양한 상황에서의 노드 통신





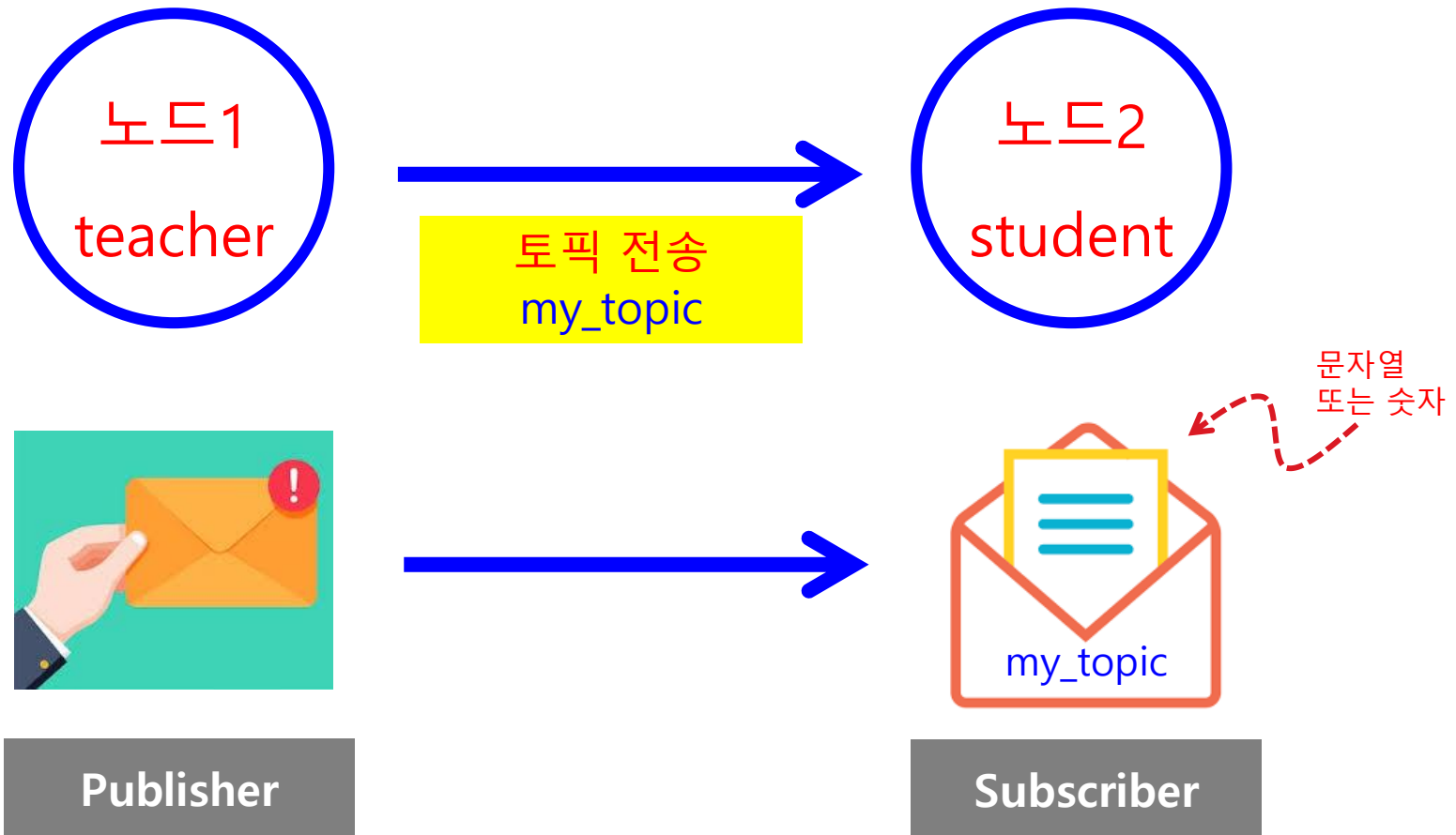
ROS 노드 통신 프로그래밍

노드 통신을 위한 패키지 만들기

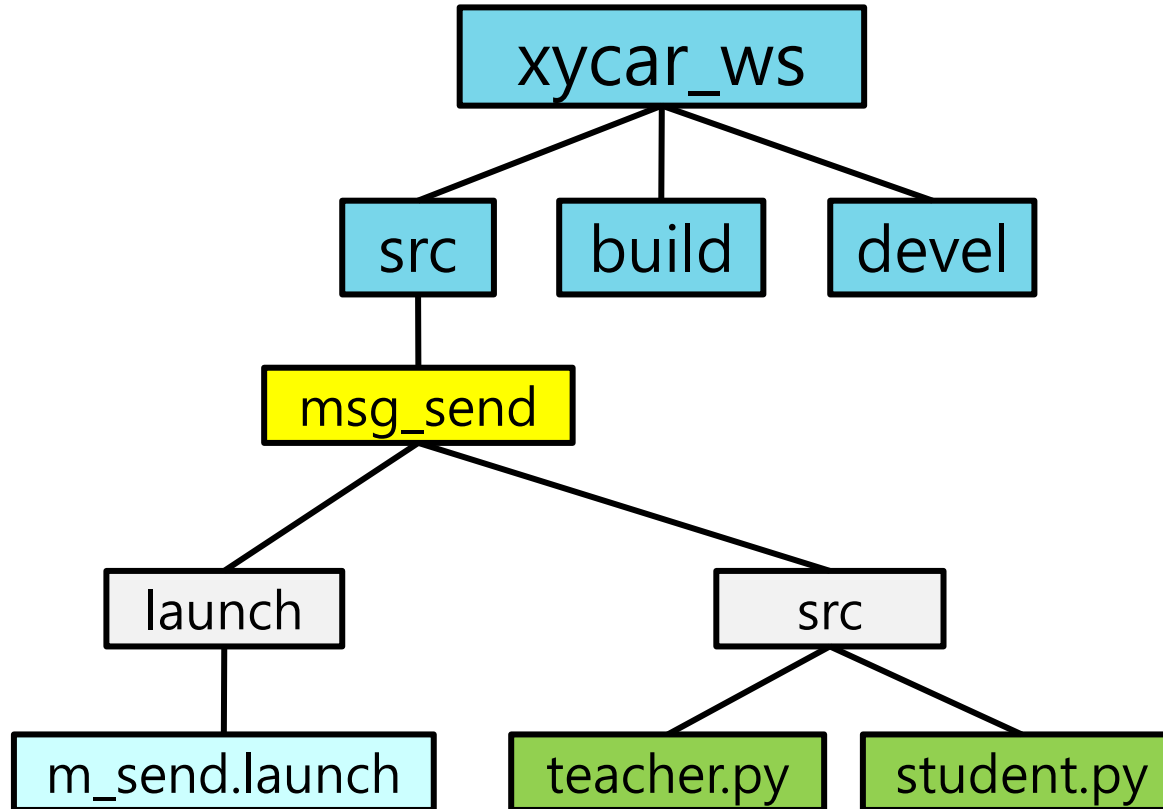


노드간 통신 개요

- 전체 구성



- 디렉토리 구조

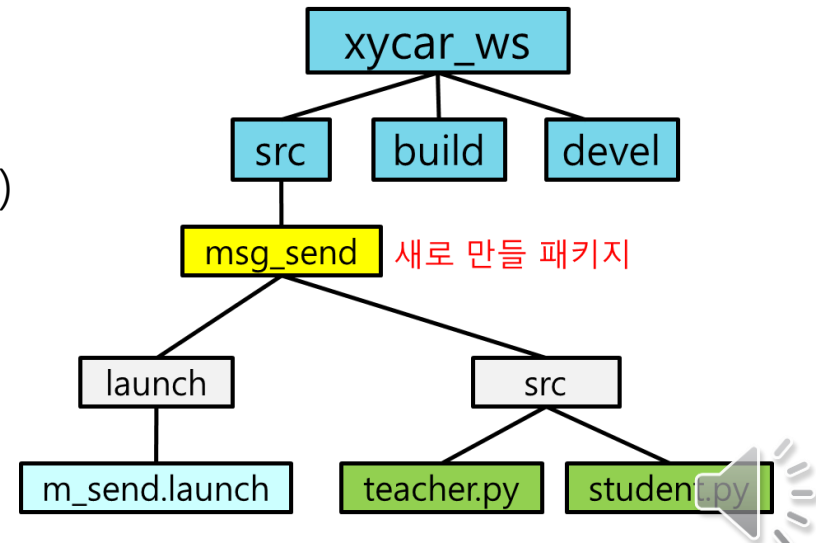


패키지 만들기

- 패키지를 담을 디렉토리로 이동
 - \$ cd ~/xycar_ws/src
- 패키지 새로 만들기
 - \$ catkin_create_pkg msg_send std_msgs rospy

{ msg_send } { std_msgs rospy }
 패키지 이름 이 패키지가 의존하고 있는
 다른 패키지들을 나열

- Launch 디렉토리 만들기
 - \$ mkdir launch (msg_send 아래에 만든다)
- 새로 만든 패키지를 빌드
 - \$ cm



파이썬 프로그램 코딩

- 토픽을 발행하고 구독하는 예제 파이썬 코드
 - 토픽 이름은 'my_topic'
 - 'teacher.py' – Publisher : 토픽에 "call me please" 담아서 전송
 - 'student.py' - Subscriber : 토픽 받아서 내용을 꺼내서 화면에 출력

teacher.py

```
1  #!/usr/bin/env python
2
3  import rospy
4  from std_msgs.msg import String
5
6  rospy.init_node('teacher')
7
8  pub = rospy.Publisher('my_topic', String)
9
10 rate = rospy.Rate(2)
11
12 while not rospy.is_shutdown():
13     pub.publish('call me please')
14     rate.sleep()
```

student.py

```
1  #!/usr/bin/env python
2
3  import rospy
4  from std_msgs.msg import String
5
6  def callback(msg):
7      print msg.data
8
9  rospy.init_node('student')
10
11 sub = rospy.Subscriber('my_topic',String,callback)
12
13 rospy.spin()
14
```

* 복사할 때 따옴표(" ") 한글/영문 자동변환 조심



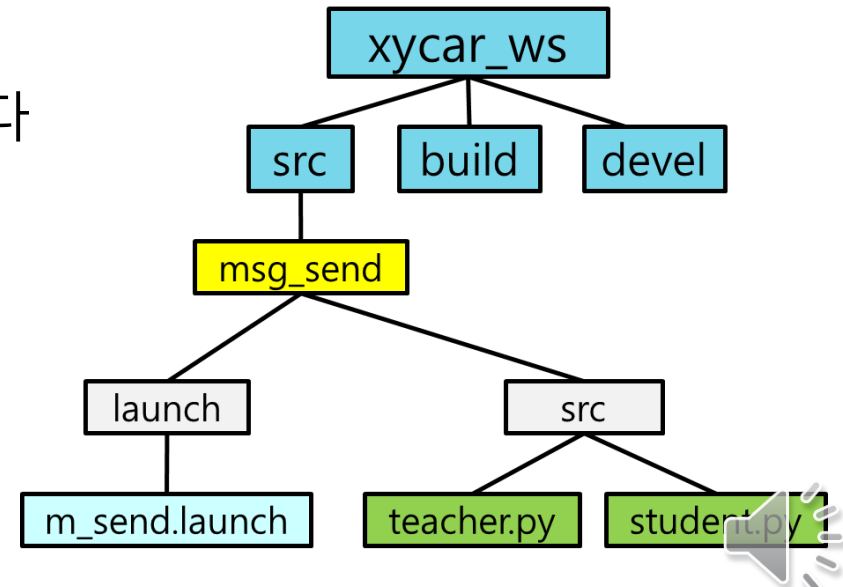
Launch 파일 만들고 실행하기

- Launch 파일 작성
 - \$ gedit m_send.launch

* 복사할 때 따옴표(' ') 한글/영문 자동변환 조심

```
<launch>
  <node pkg="msg_send" type="teacher.py" name="teacher"/>
  <node pkg="msg_send" type="student.py" name="student" output="screen"/>
</launch>
```

- \$ cm (빌드한다)
- 파이썬 파일에는 실행권한을 부여해야 한다
 - \$ chmod +x teacher.py student.py
- Launch 파일 실행 방법
 - \$ roslaunch msg_send m_send.launch



실행 결과

- \$ roslaunch msg_send m_send.launch

\$ rqt_graph

```
nvidia@tegra-ubuntu: ~/xycar_ws/src/msg_send/s
auto-starting new master
process[master]: started with pid [4387]
ROS_MASTER_URI=http://localhost:11311

setting /run_id to 3816b9f0-351
process[rosout-1]: started with pid [4388]
started core service [/rosout]
process[teacher-2]: started with pid [4389]
process[student-3]: started with pid [4390]
/home/nvidia/xycar_ws/src/msg_send/s
r should be created with an exp
http://wiki.ros.org/rospy/Overvi
ation.
pub = rospy.Publisher('my_top
call me please
call me please
call me please
call me please
call me please
call me please
call me please
call me please
```

Node Graph

Nodes/Topics (all) / /

Group: 2 Namespaces ☒ Actions ☒ tf ☒ Images ☒ Highlight ☒ Fit

Hide: ☒ Dead sinks ☒ Leaf topics ☒ Debug ☐ tf ☒ Unreachable ☒ Params

/teacher → /my_topic → /student



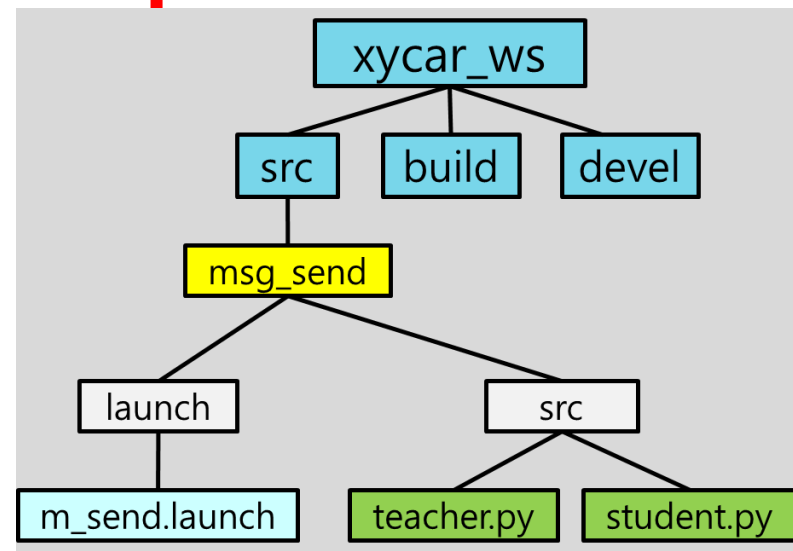
실습 (간단한 노드 통신 구현)



정리하면

• 전체 작업 과정

- `$ cd ~/xycar_ws/src`
- `$ catkin_create_pkg msg_send std_msgs rospy`
- `$ mkdir launch`
- `$ cd ~/xycar_ws/src/msg_send/src`
- `$ gedit student.py`
- `$ gedit teacher.py`
- `$ chmod +x student.py teacher.py`
- `$ cd ~/xycar_ws/src/msg_send/launch`
- `$ gedit m_send.launch`
- `$ cm`
- `$ roslaunch msg_send m_send.launch`



ROS 노드 통신 프로그래밍

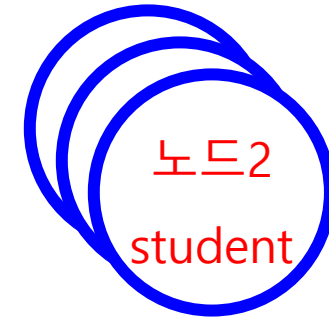
1:N, N:1, N:N 통신



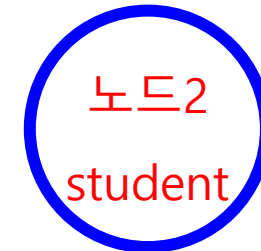
다양한 통신 시나리오

- 다양한 구성

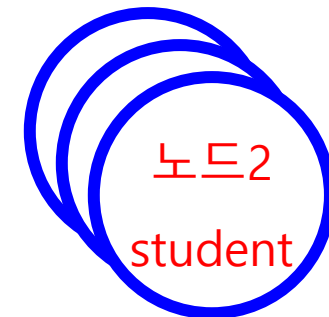
1:N 통신



N:1 통신

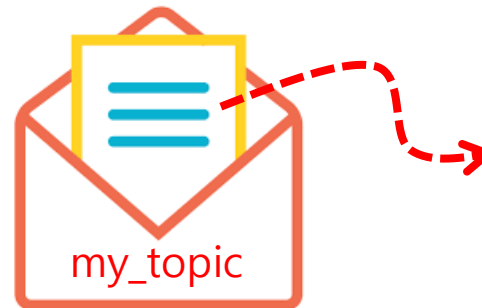


N:N 통신



토픽 메시지 타입 변경

- String 대신에 Int32 사용



String 대신
Int32 사용하자.

Count 값을
넣어서 보내보자.
1, 2, 3, 4, ...

Int32 타입의 메시지를 주고 받는 파이썬 코드

- \$ gedit teacher_int.py
- \$ gedit student_int.py

teacher_int.py

```
#!/usr/bin/env python

import rospy
from std_msgs.msg import Int32

rospy.init_node('teacher')

pub = rospy.Publisher('my_topic', Int32)

rate = rospy.Rate(2)
count = 1

while not rospy.is_shutdown():
    pub.publish(count)
    count = count+1
    rate.sleep()
```

student_int.py

```
#!/usr/bin/env python

import rospy
from std_msgs.msg import Int32

def callback(msg):
    print msg.data

rospy.init_node('student')

sub = rospy.Subscriber('my_topic', Int32, callback)

rospy.spin()
```



노드를 여러 개 띄울 때

- 하나의 코드로 여러 개의 노드를 연결하려면 각 노드의 이름을 달리해야
 - 노드의 init 함수에서 `anonymous=True` 값을 넣어주면 노드이름이 자동설정됨
 - 예시 `rospy.init_node('student', anonymous = True)`

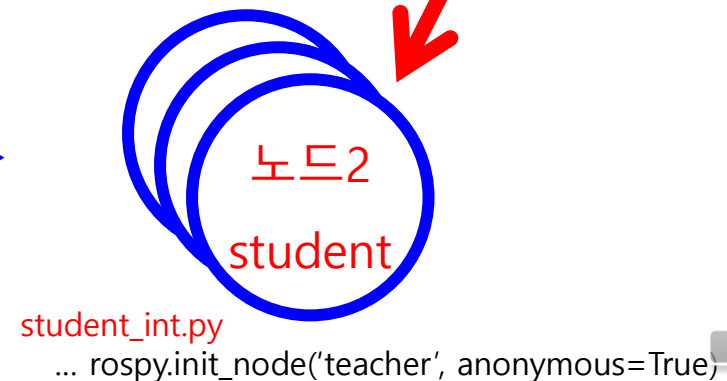
```
$ rosrun msg_send teacher_int-1.py
$ rosrun msg_send teacher_int-2.py
$ rosrun msg_send teacher_int-3.py
```

```
$ rosrun msg_send student_int.py
$ rosrun msg_send student_int.py
$ rosrun msg_send student_int.py
```

```
teacher_int-1.py
... rospy.init_node('teacher111')
```

```
teacher_int-2.py
... rospy.init_node('teacher222')
```

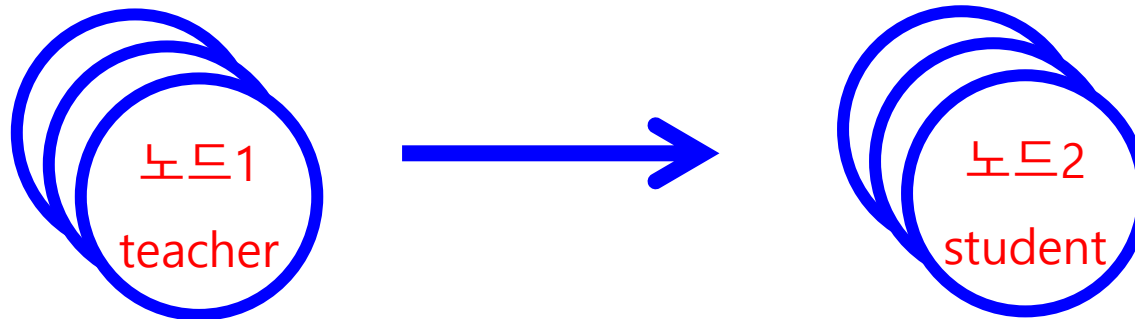
```
teacher_int-3.py
... rospy.init_node('teacher333')
```



노드를 여러 개 띄울 때

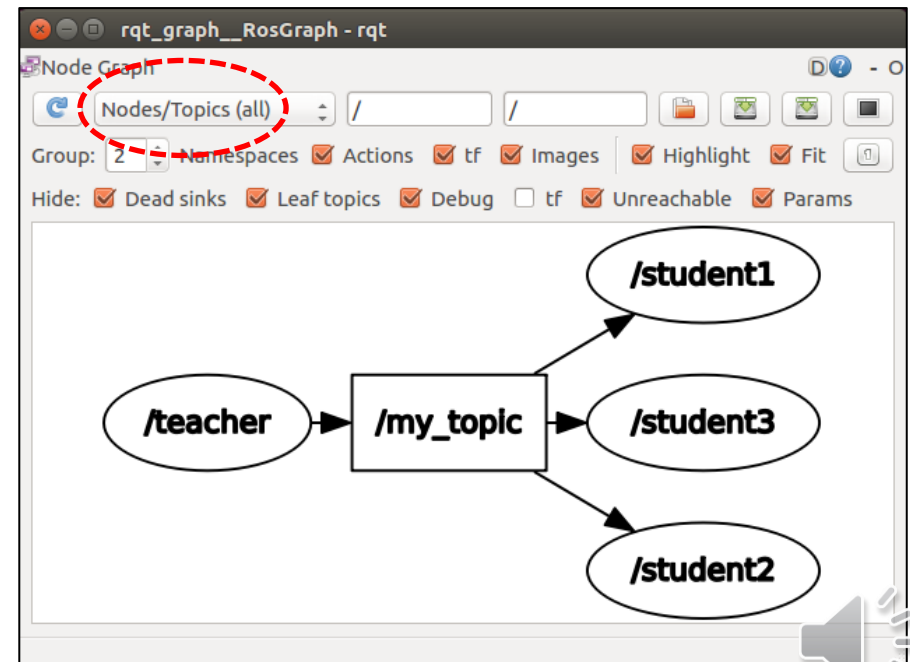
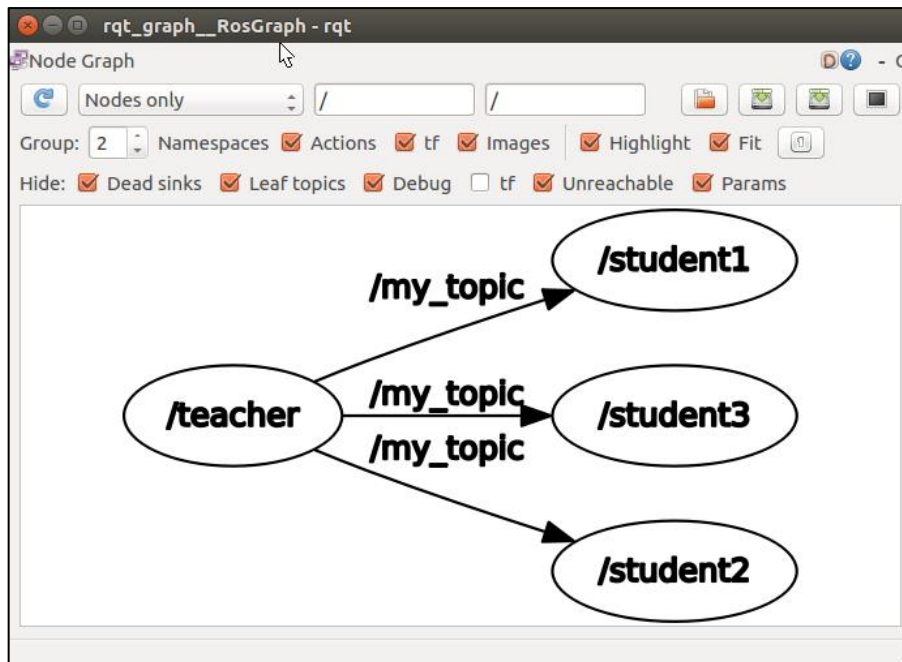
- Launch 파일을 이용해서 roslaunch 명령으로 여러 노드를 띄울 수 있음
 - 노드 설정에서 name="ooo" 부분을 다르게 설정
 - 예시 <node pkg="msg_send" type="teacher_int.py" name="teacher1"/>

```
<launch>
  <node pkg="msg_send" type="teacher_int.py" name="teacher1"/>
  <node pkg="msg_send" type="teacher_int.py" name="teacher2"/>
  <node pkg="msg_send" type="teacher_int.py" name="teacher3"/>
  <node pkg="msg_send" type="student_int.py" name="student1" output="screen"/>
  <node pkg="msg_send" type="student_int.py" name="student2" output="screen"/>
  <node pkg="msg_send" type="student_int.py" name="student3" output="screen"/>
</launch>
```



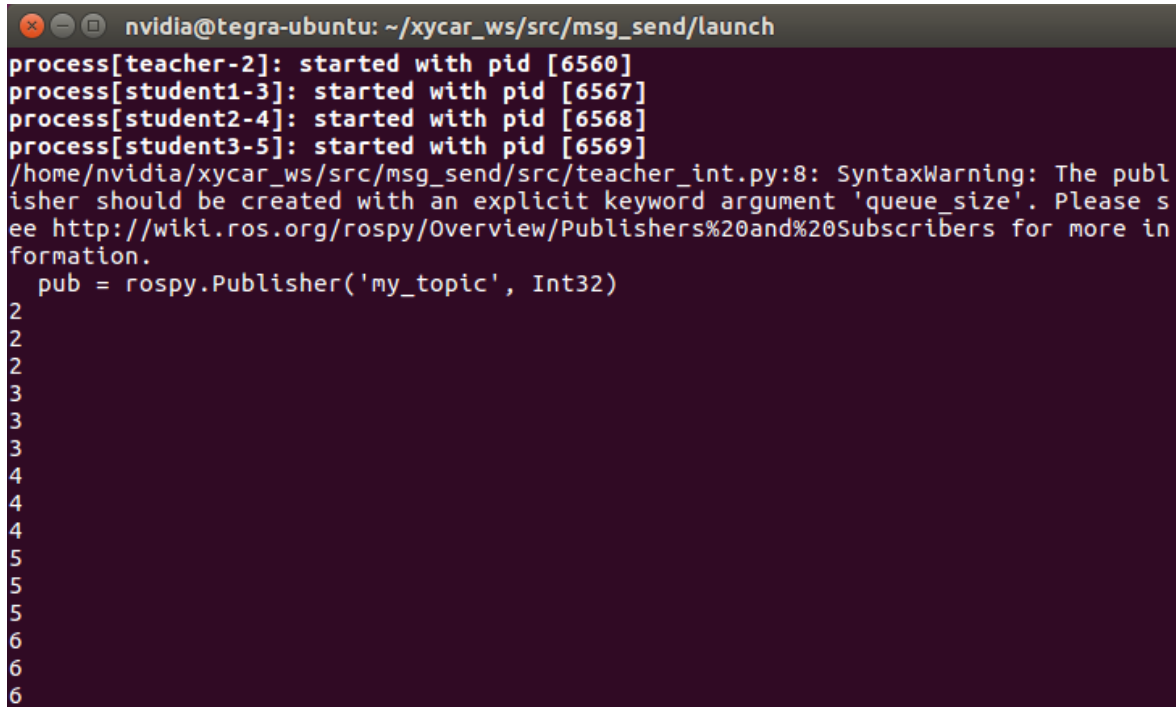
- Launch 파일만 바꿔서 (m_send_1n.launch)

```
<launch>
  <node pkg="msg_send" type="teacher_int.py" name="teacher"/>
  <node pkg="msg_send" type="student_int.py" name="student1" output="screen"/>
  <node pkg="msg_send" type="student_int.py" name="student2" output="screen"/>
  <node pkg="msg_send" type="student_int.py" name="student3" output="screen"/>
</launch>
```



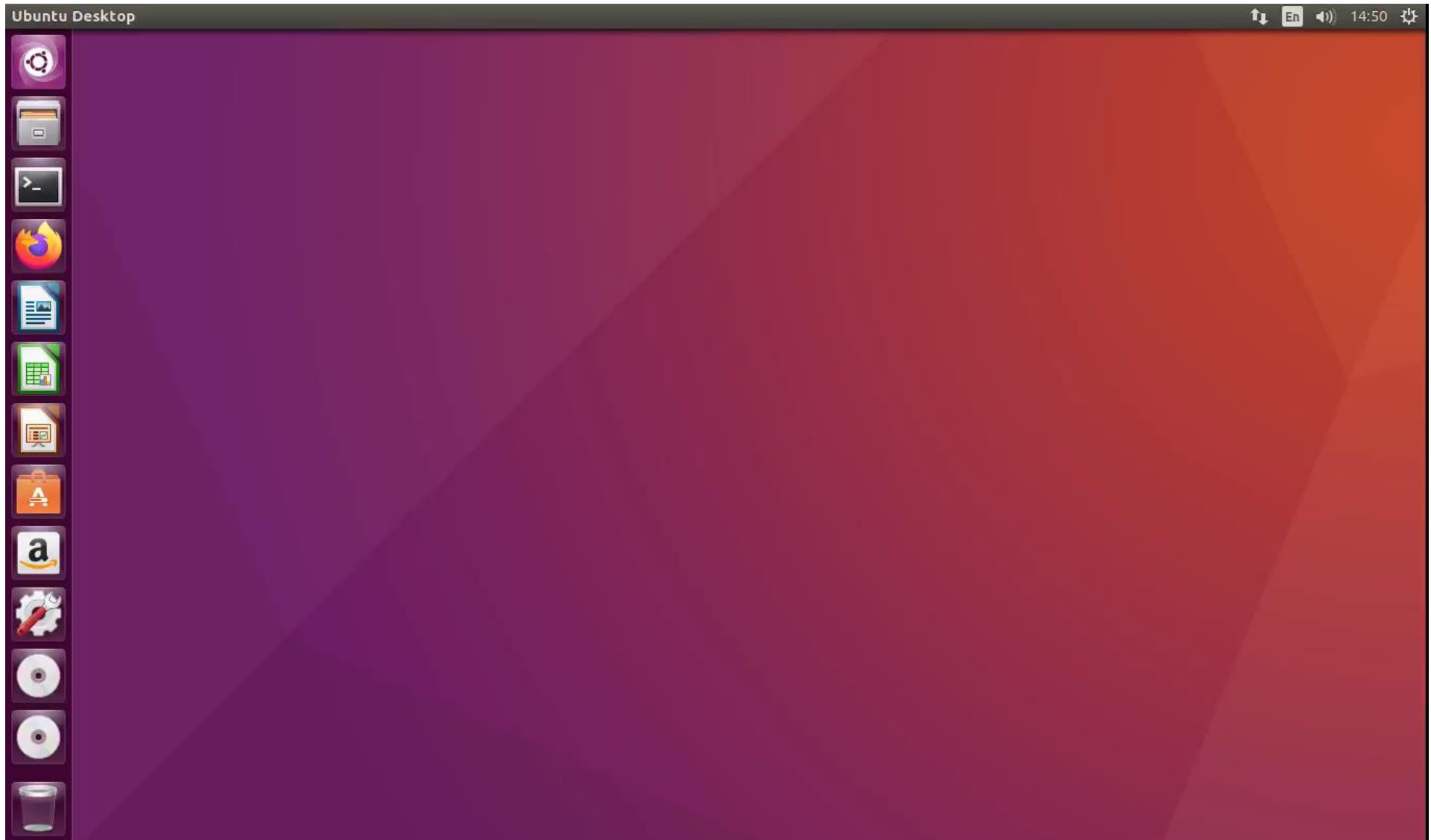
- Launch 파일만 바꿔서 (m_send_1n.launch)

```
<launch>
  <node pkg="msg_send" type="teacher_int.py" name="teacher"/>
  <node pkg="msg_send" type="student_int.py" name="student1" output="screen"/>
  <node pkg="msg_send" type="student_int.py" name="student2" output="screen"/>
  <node pkg="msg_send" type="student_int.py" name="student3" output="screen"/>
</launch>
```



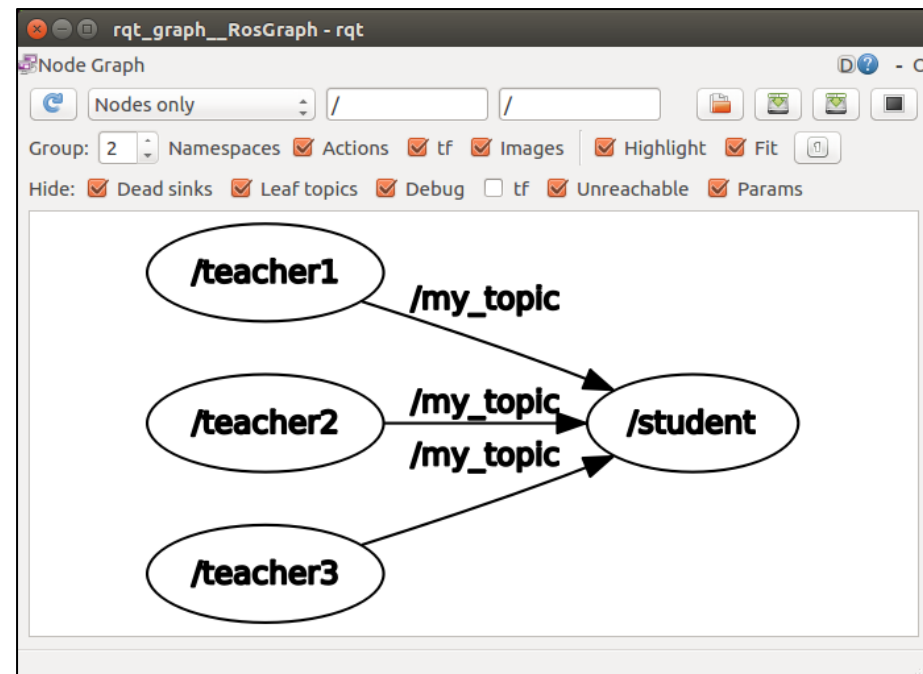
```
nvidia@tegra-ubuntu: ~/xycar_ws/src/msg_send/launch
process[teacher-2]: started with pid [6560]
process[student1-3]: started with pid [6567]
process[student2-4]: started with pid [6568]
process[student3-5]: started with pid [6569]
/home/nvidia/xycar_ws/src/msg_send/src/teacher_int.py:8: SyntaxWarning: The publisher should be created with an explicit keyword argument 'queue_size'. Please see http://wiki.ros.org/rospy/Overview/Publishers%20and%20Subscribers for more information.
  pub = rospy.Publisher('my_topic', Int32)
2
2
2
3
3
3
4
4
4
5
5
5
6
6
6
```

실습 (1:N 통신)



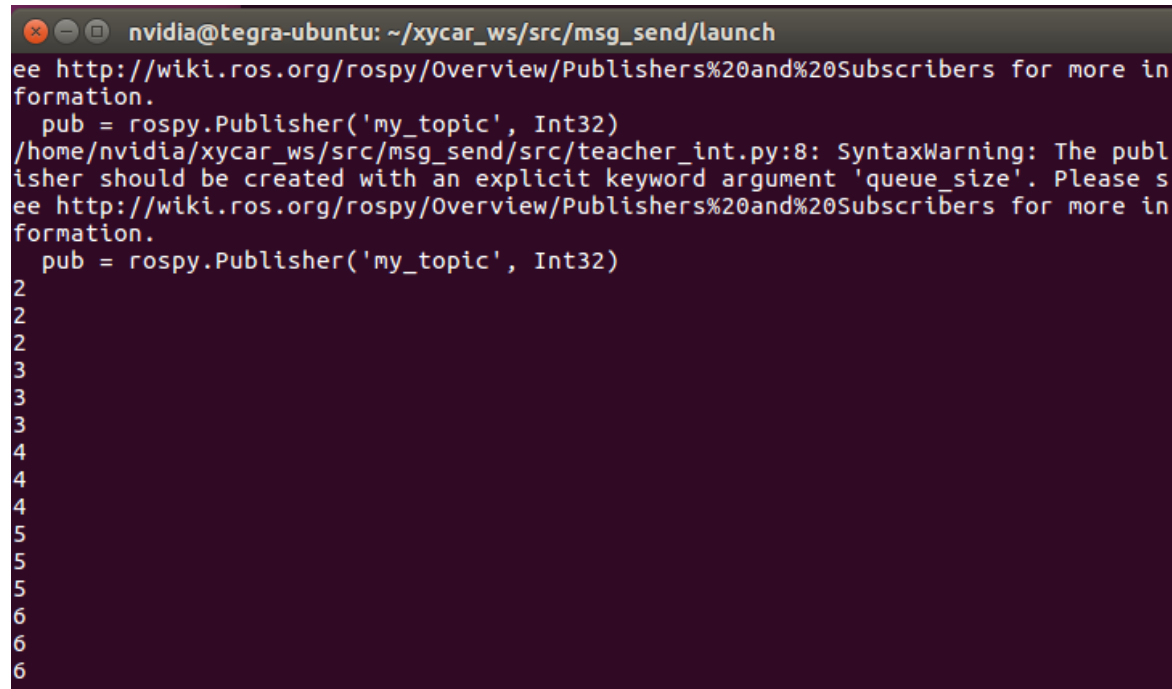
- Launch 파일만 바꿔서 (m_send_n1.launch)

```
<launch>
  <node pkg="msg_send" type="teacher_int.py" name="teacher1"/>
  <node pkg="msg_send" type="teacher_int.py" name="teacher2"/>
  <node pkg="msg_send" type="teacher_int.py" name="teacher3"/>
  <node pkg="msg_send" type="student_int.py" name="student" output="screen"/>
</launch>
```



- Launch 파일만 바꿔서 (m_send_n1.launch)

```
<launch>
  <node pkg="msg_send" type="teacher_int.py" name="teacher1"/>
  <node pkg="msg_send" type="teacher_int.py" name="teacher2"/>
  <node pkg="msg_send" type="teacher_int.py" name="teacher3"/>
  <node pkg="msg_send" type="student_int.py" name="student" output="screen"/>
</launch>
```

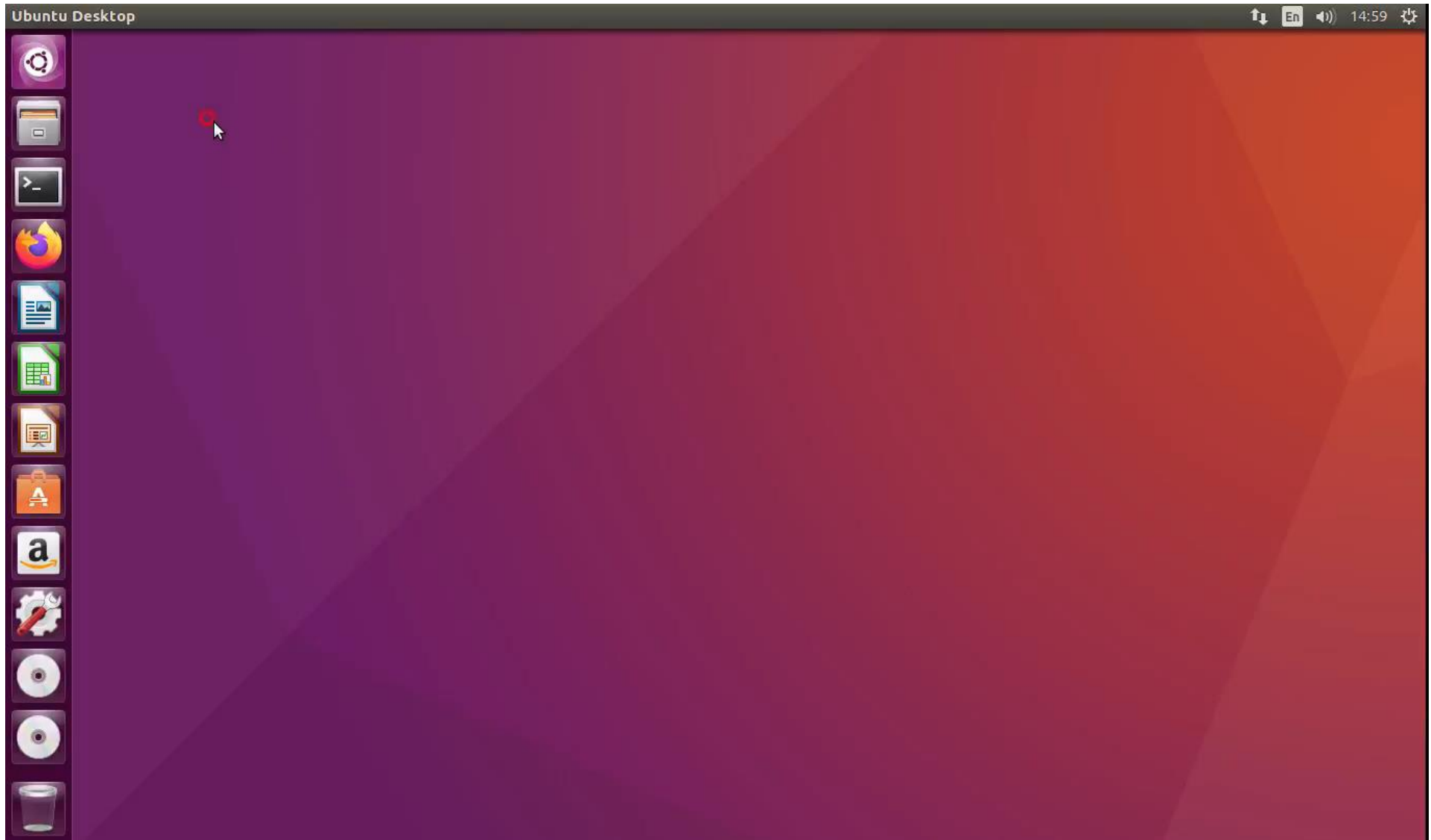


nvidia@tegra-ubuntu: ~/xycar_ws/src/msg_send/launch

```
ee http://wiki.ros.org/rospy/Overview/Publishers%20and%20Subscribers for more in
formation.
  pub = rospy.Publisher('my_topic', Int32)
/home/nvidia/xycar_ws/src/msg_send/src/teacher_int.py:8: SyntaxWarning: The publ
isher should be created with an explicit keyword argument 'queue_size'. Please s
ee http://wiki.ros.org/rospy/Overview/Publishers%20and%20Subscribers for more in
formation.
  pub = rospy.Publisher('my_topic', Int32)
2
2
2
3
3
3
4
4
4
5
5
5
6
6
6
```

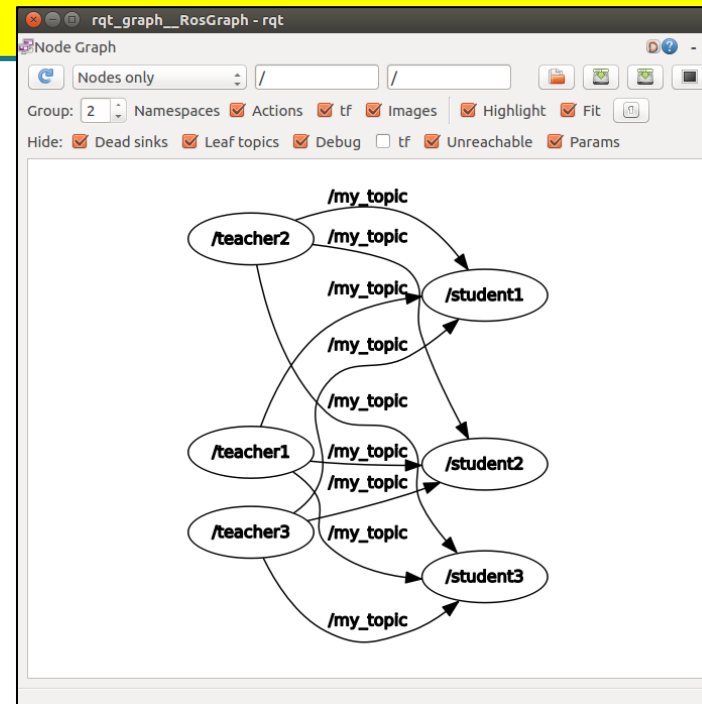


실습 (N:1 통신)



- Launch 파일만 바꿔서 (m_send_nn.launch)

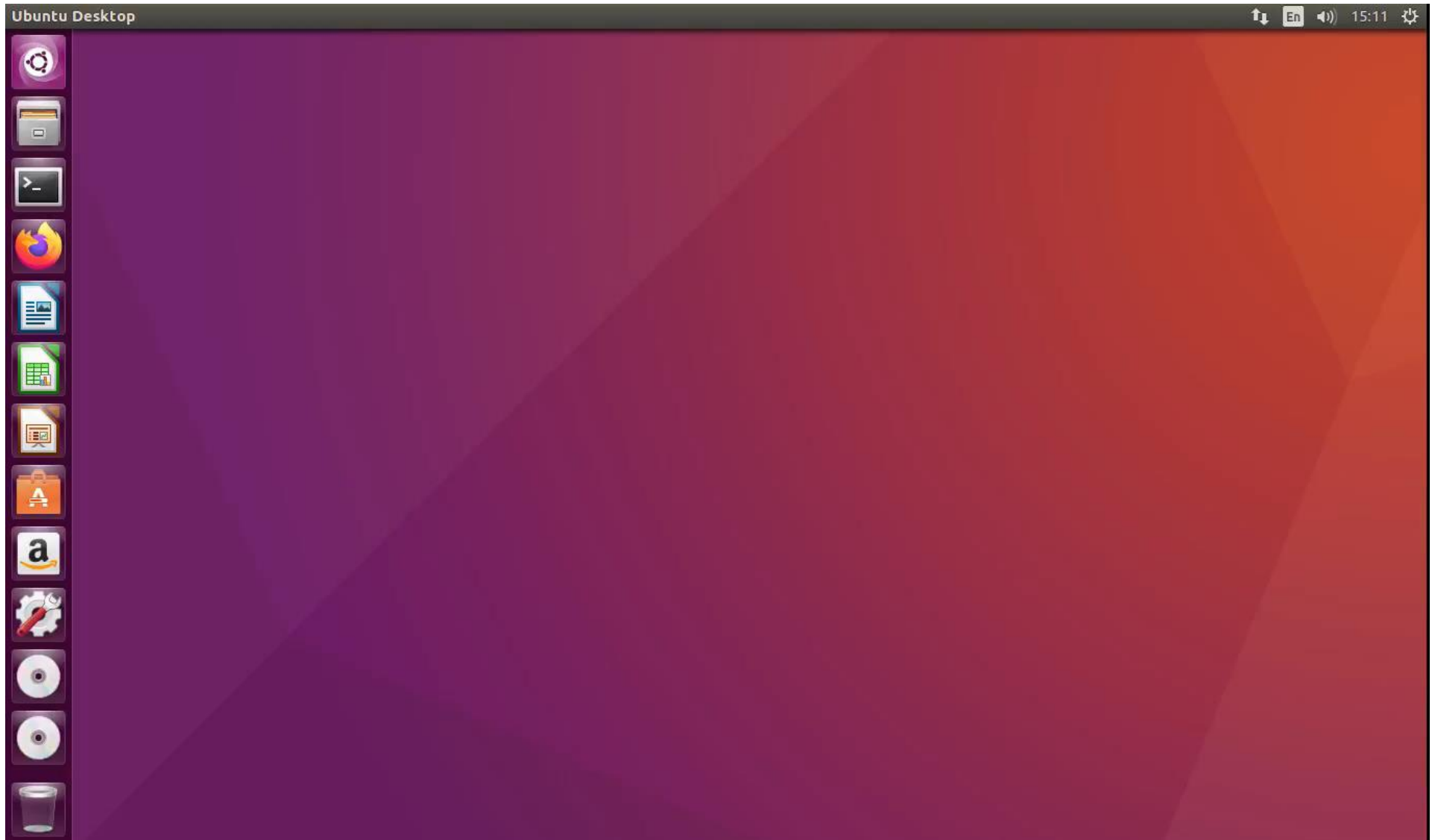
```
<launch>
  <node pkg="msg_send" type="teacher_int.py" name="teacher1"/>
  <node pkg="msg_send" type="teacher_int.py" name="teacher2"/>
  <node pkg="msg_send" type="teacher_int.py" name="teacher3"/>
  <node pkg="msg_send" type="student_int.py" name="student1" output="screen"/>
  <node pkg="msg_send" type="student_int.py" name="student2" output="screen"/>
  <node pkg="msg_send" type="student_int.py" name="student3" output="screen"/>
</launch>
```



- ```
<launch>
 <node pkg="msg_send" type="teacher_int.py" name="teacher1"/>
 <node pkg="msg_send" type="teacher_int.py" name="teacher2"/>
 <node pkg="msg_send" type="teacher_int.py" name="teacher3"/>
 <node pkg="msg_send" type="student_int.py" name="student1" output="screen"/>
 <node pkg="msg_send" type="student_int.py" name="student2" output="screen"/>
 <node pkg="msg_send" type="student_int.py" name="student3" output="screen"/>
</launch>
```



# 실습 (N:N 통신)

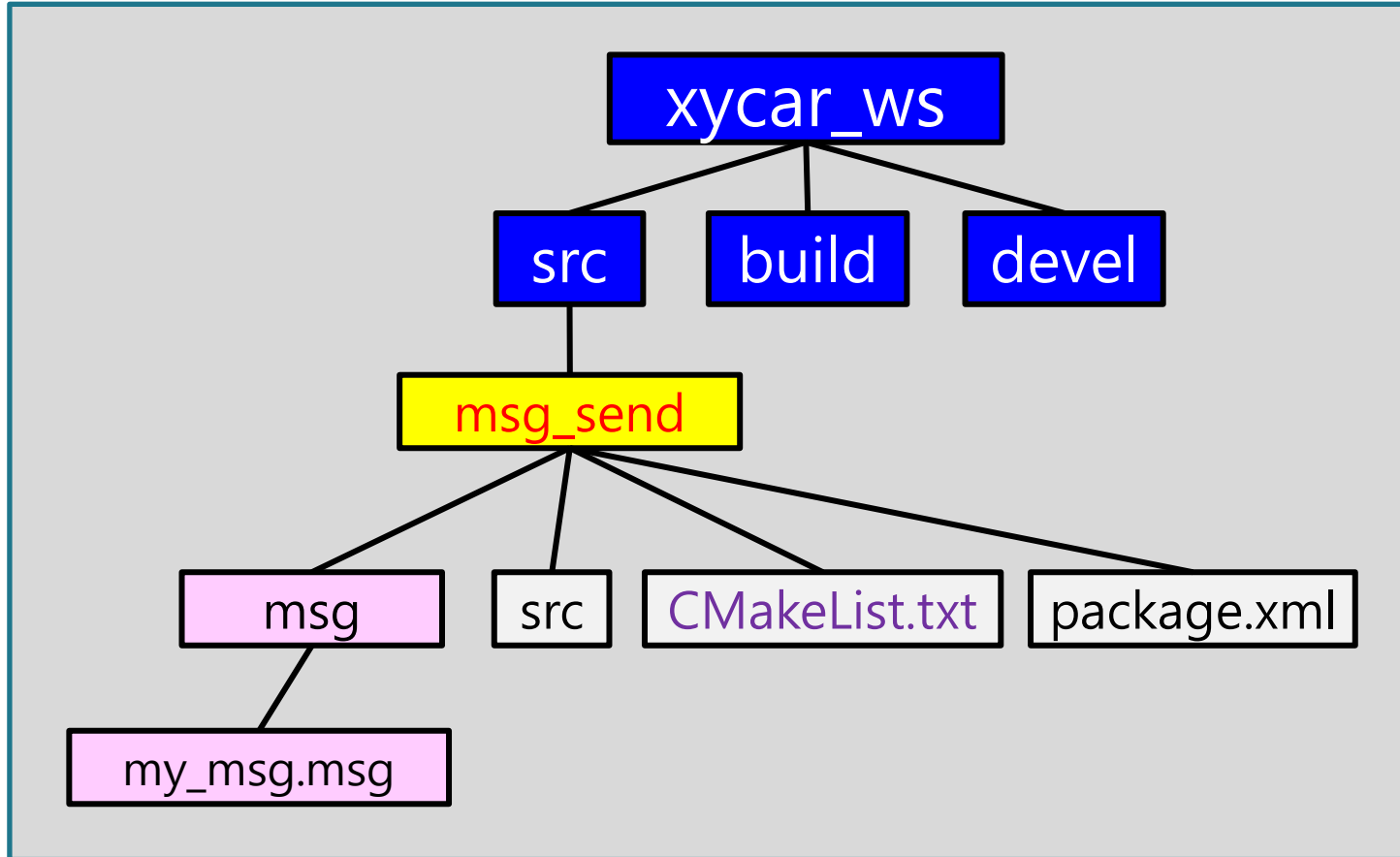




# ROS 노드 통신 프로그래밍

나만의 메시지 만들기

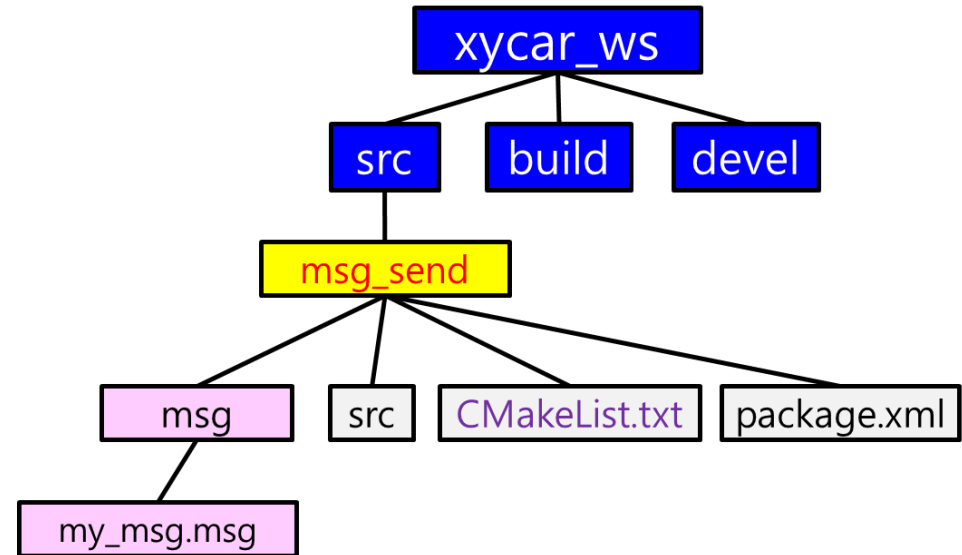




# Custom Message 사용 방법

- 메시지 파일 생성 및 작성

- \$ cd ~/xycar\_ws/src
- \$ roscd msg\_send
- \$ mkdir msg
- \$ cd msg
- \$ gedit my\_msg.msg
  - ▶ 아래 내용 작성

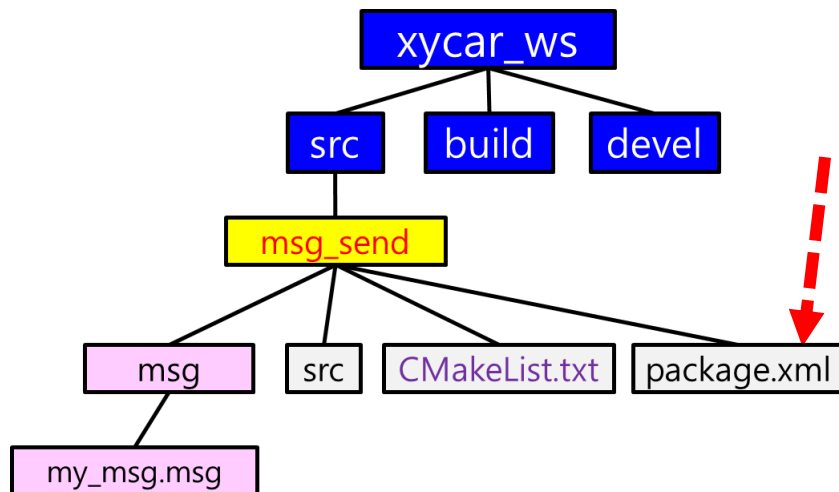


```
*my_msg.msg (~/xycar/src/msg_send/msg) - gedit
Open
1 string first_name
2 string last_name
3 int32 age
4 int32 score
5 string phone_number
6 int32 id_number
```

```
string first_name
string last_name
int32 age
int32 score
string phone_number
int32 id_number
```

# Custom Message 선언

- package.xml 수정
  - \$ gedit package.xml
  - 파일 아래 쪽에 내용 추가
  - `<build_depend>message_generation</build_depend>`
  - `<exec_depend>message_runtime</exec_depend>`



```

<buildtool_depend>catkin</buildtool_depend>
<build_depend>rospy</build_depend>
<build_depend>std_msgs</build_depend>
<build_export_depend>rospy</build_export_depend>
<build_export_depend>std_msgs</build_export_depend>
<exec_depend>rospy</exec_depend>
<exec_depend>std_msgs</exec_depend>
<build_depend>message_generation</build_depend>
<exec_depend>message_runtime</exec_depend>

```

2줄 새롭게  
추가

```

<!-- The export tag contains other, unspecified, tags -->
<export>
 <!-- Other tools can request additional information be placed here -->

</export>
</package>

```



# Custom Message 선언

- CMakeLists.txt 수정 (# 코멘트 삭제하고, 추가 삽입하고)

```

• find_package(catkin REQUIRED COMPONENTS
 rospy
 std_msgs
 message_generation
)

• add_message_files(
 FILES
 my_msg.msg
)

• generate_messages(
 DEPENDENCIES
 std_msgs
)

• catkin_package(
 CATKIN_DEPENDS message_runtime
 # INCLUDE_DIRS include
 # LIBRARIES my_package
 # CATKIN_DEPENDS rospy std_msgs
 # DEPENDS system_lib
)

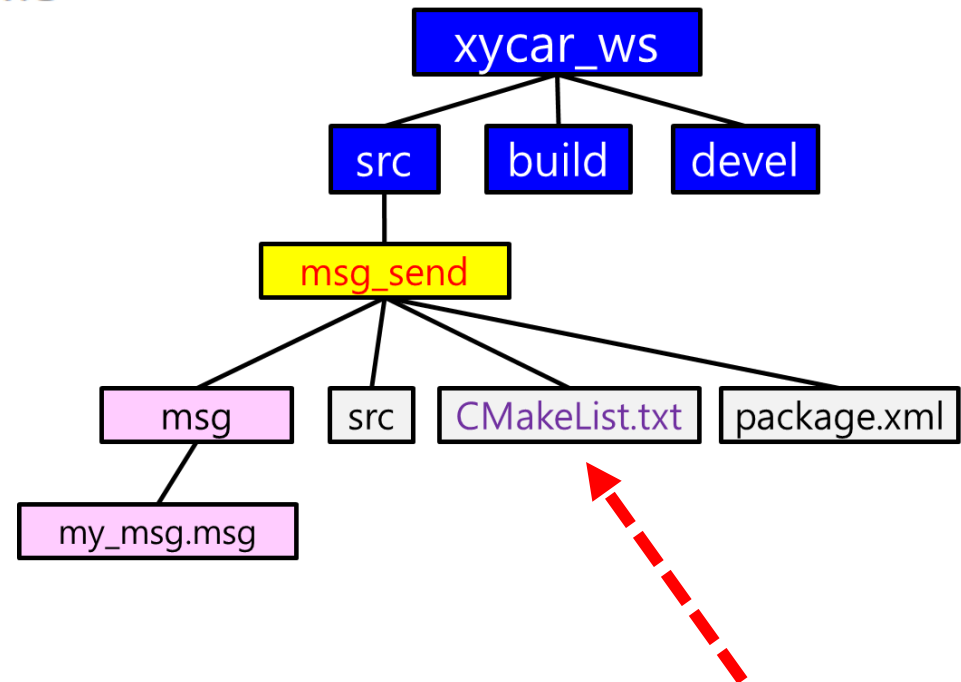
```

1줄 새롭게 추가

코멘트 풀고  
1줄 새롭게 추가

코멘트 풀기

1줄 새롭게 추가



# Custom Message 선언

- CMakeLists.txt 수정 (# 코멘트 삭제하고, 추가 삽입하고)

## 수정 전

```
...
10 find_package(catkin REQUIRED COMPONENTS
11 rospy
12 std_msgs
13)

...
49 # add_message_files(
50 # FILES
51 # Message1.msg
52 # Message2.msg
53 #)

...
70 # generate_messages(
71 # DEPENDENCIES
72 # std_msgs
73 #)

...
104 catkin_package(
105 # INCLUDE_DIRS include
106 # LIBRARIES my_pkg1
107 # CATKIN_DEPENDS rospy std_msgs
108 # DEPENDS system_lib
109)
```

## 수정 완료

```
...
10 find_package(catkin REQUIRED COMPONENTS
11 rospy
12 std_msgs
13 message_generation
14)

...
49 add_message_files(
50 FILES
51 my_msg.msg
52 # Message1.msg
53 # Message2.msg
54)

...
70 generate_messages(
71 DEPENDENCIES
72 std_msgs
73)

...
104 catkin_package(
105 CATKIN_DEPENDS message_runtime
106 # INCLUDE_DIRS include
107 # LIBRARIES my_pkg1
108 # CATKIN_DEPENDS rospy std_msgs
109 # DEPENDS system_lib
110)
```

1줄 추가

코멘트 제거

1줄 추가

코멘트 제거

1줄 추가





# Custom Message 설정과 확인

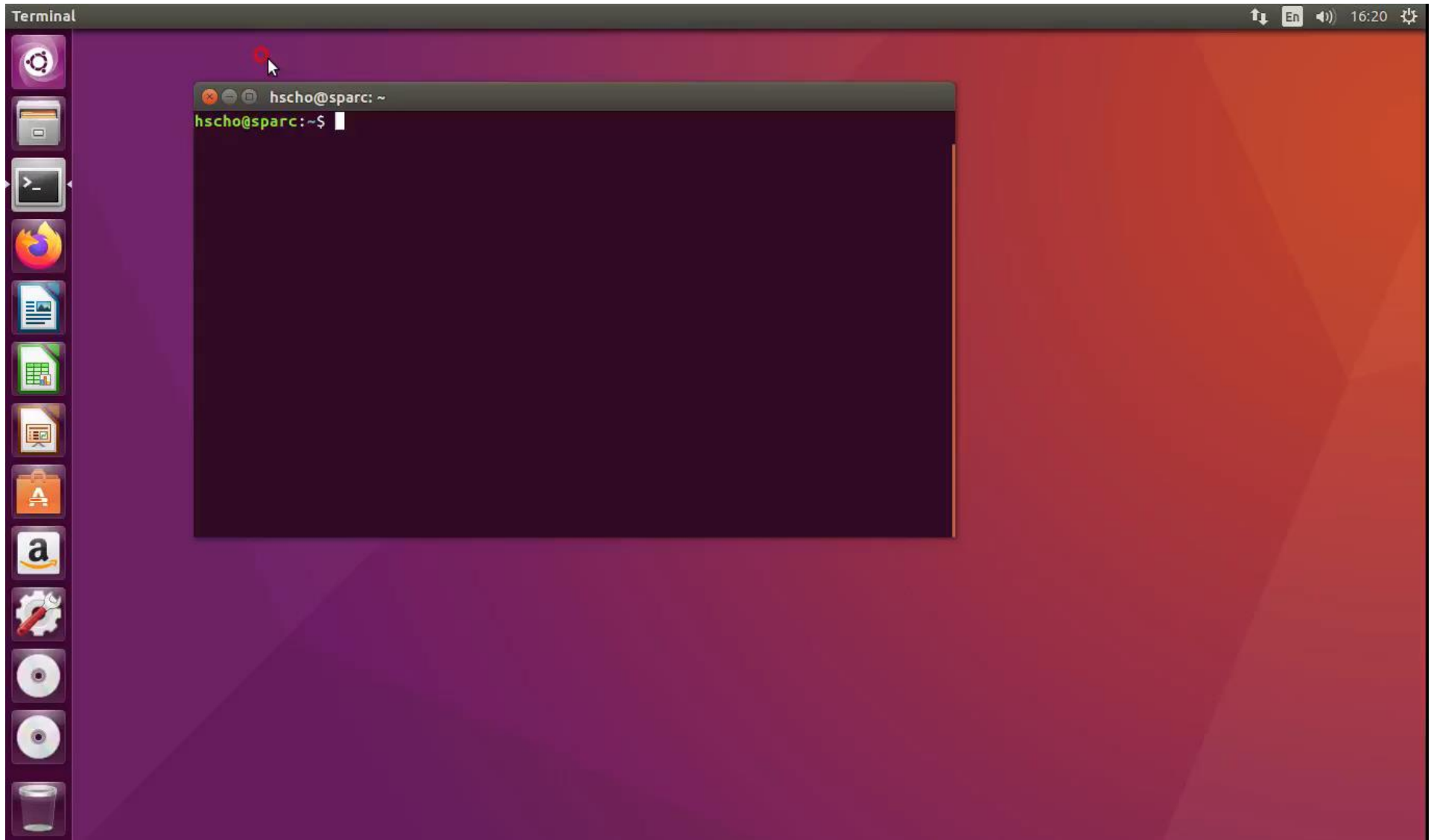
- \$ cm
- \$ rosmmsg show **my\_msg** 으로 결과 확인

```
Terminal
xycar:~/xycar/src/msg_send$ rosmmsg show msg_send/my_msg
string first_name
string last_name
int32 age
int32 score
string phone_number
int32 id_number

xycar:~/xycar/src/msg_send$ rosmmsg show my_msg
[msg_send/my_msg]:
string first_name
string last_name
int32 age
int32 score
string phone_number
int32 id_number
```

- 아래의 디렉토리에서 생성된 msg 확인 가능함
  - (파이썬) ~/xycar\_ws/devel/lib/python2.7/dist\_packages/**msg\_send**/msg

# 실습 (나만의 메시지 만들기)



# 내 코드 안에서 Custom Message 사용하기

---

- 코드 안에 include or import 하는 법
  - (파이썬) `from msg_send.msg import my_msg`
- 다른 패키지에서도 custom msg 사용 가능
- 참고 링크
  - <http://wiki.ros.org/ROS/Tutorials/CreatingMsgAndSrv>

# my\_msg 사용 예제

- 샘플 파이썬 코드 : 메시지 발행 Publisher 노드
  - \$ gedit msg\_sender.py

```
#!/usr/bin/env python

import rospy
from msg_send.msg import my_msg

rospy.init_node('msg_sender', anonymous=True)
pub = rospy.Publisher('msg_to_xycar', my_msg)

msg = my_msg()
msg.first_name = "gildon"
msg.last_name = "Hong"
msg.id_number = 20041003
msg.phone_number = "010-8990-3003"

rate = rospy.Rate(1)
while not rospy.is_shutdown():
 pub.publish(msg)
 print("sending message")
 rate.sleep()
```

my\_msg 사용

my\_msg 안에  
데이터 채워 넣기

my\_msg 토픽  
발행하기



# my\_msg 사용 예제

- 샘플 파이썬 코드 – 메시지 구독 Subscriber 노드
  - \$ gedit msg\_receiver.py

```
#!/usr/bin/env python
```

```
import rospy
from msg_send.msg import my_msg
```

my\_msg 사용

```
def callback(msg):
 print("1. Name : ", msg.last_name + msg.first_name)
 print("2. ID : ", msg.id_number)
 print("3. Phone Number : ", msg.phone_number)
```

my\_msg  
데이터 꺼내기

```
rospy.init_node('msg_receiver', anonymous=True)
```

```
sub = rospy.Subscriber('msg_to_xycar', my_msg, callback)
```

my\_msg 토픽  
구독하기

```
rospy.spin()
```

- \$ chmod +x msg\_sender.py msg\_receiver.py

# my\_msg 사용 예제의 실행 결과

- 빌드 & 실행
  - \$ cm
  - \$ roscore
  - \$ rosrund msg\_send msg\_receiver.py
  - \$ rosrund msg\_send msg\_sender.py

```
sungmin@machine: ~/catkin_ws
sungmin@machine:~/catkin_ws$ rosrund msg_send msg_sender.py
sending message
sending message
sending message
sending message
sending message
sending message
sending message
sending message
sending message
sending message
sending message
```

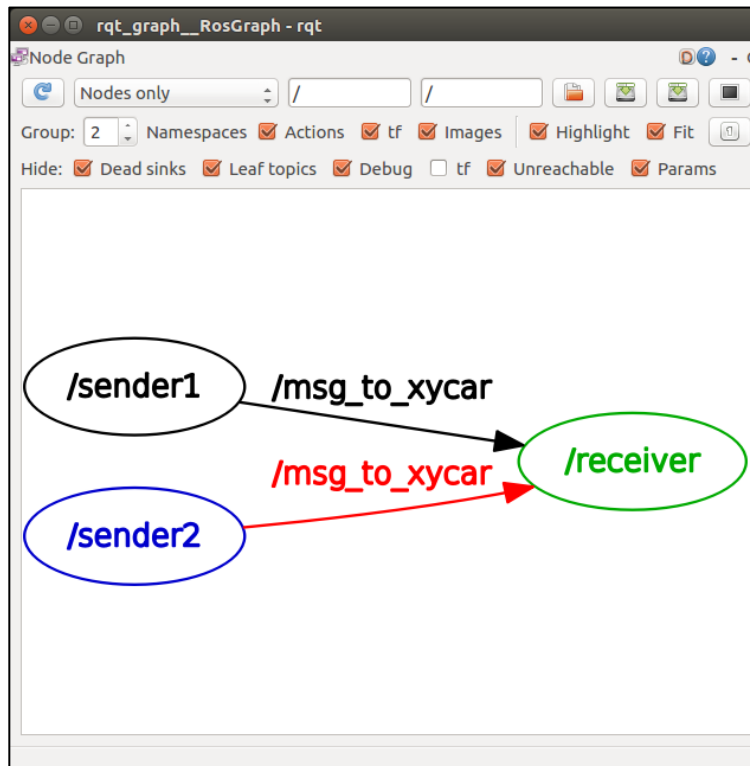
```
sungmin@machine: ~
sungmin@machine:~$ rosrund msg_send msg_receiver.py
('1. Name : ', 'Honggildon')
('2. ID : ', 20041003)
('3. Phone Number : ', '010-8990-3003')
('1. Name : ', 'Honggildon')
('2. ID : ', 20041003)
('3. Phone Number : ', '010-8990-3003')
('1. Name : ', 'Honggildon')
('2. ID : ', 20041003)
('3. Phone Number : ', '010-8990-3003')
('1. Name : ', 'Honggildon')
('2. ID : ', 20041003)
('3. Phone Number : ', '010-8990-3003')
^Csungmin@machine:~$
```



# my\_msg 사용 예제의 실행 결과

- \$ roslaunch msg\_send m\_send\_sr.launch

```
<launch>
 <node pkg="msg_send" type="msg_sender.py" name="sender1"/>
 <node pkg="msg_send" type="msg_sender.py" name="sender2"/>
 <node pkg="msg_send" type="msg_receiver.py" name="receiver" output="screen"/>
</launch>
```



```
sungmin@machine: ~/catkin_ws
('2. ID : ', 20041003)
('3. Phone Number : ', '010-8990-3003')
('1. Name : ', 'Honggildon')
('2. ID : ', 20041003)
('3. Phone Number : ', '010-8990-3003')
('1. Name : ', 'Honggildon')
('2. ID : ', 20041003)
('3. Phone Number : ', '010-8990-3003')
('1. Name : ', 'Honggildon')
('2. ID : ', 20041003)
('3. Phone Number : ', '010-8990-3003')
('1. Name : ', 'Honggildon')
('2. ID : ', 20041003)
('3. Phone Number : ', '010-8990-3003')
('1. Name : ', 'Honggildon')
```



# 실습 (나만의 메시지 사용 예제의 실행)

The screenshot shows a web browser displaying the ROS Wiki page "ROS/Tutorials/CreatingMsgAndSrv". The page content includes instructions on ensuring the `generate_messages` macro is used correctly for ROS Hydro and later versions. A terminal window is overlaid on the page, showing the execution of the `rosmake` command to generate Python files from a custom message definition.

By adding the `generate_messages` macro manually, we make sure that CMake knows when it has to reconfigure the project and other `.msg` files.

Now we must ensure the `generate_messages` macro is used correctly. For ROS Hydro and later, you need to use the `generate_messages` macro in the `CMakeLists.txt` file.

```
generate_messages (
DEPENDENCIES
std_msgs
)
```

so it looks like:

```
generate_messages (
 DEPENDENCIES
 std_msgs
)
```

In earlier versions, you may just need to use the `generate_messages` macro in the `CMakeLists.txt` file.

```
generate_messages ()
```

Now you're ready to generate source files from your msg definition. If you want to do so right now, skip to the next section: [Common step for msg and srv.](#)

Terminal Output:

```
hscho@sparc: ~/xycar_ws
[85%] Generating Python from MSG msg_send/my_msg
[100%] Generating Python msg __init__.py for msg_send
[100%] Built target msg_send_generate_messages_py
Scanning dependencies of target msg_send_generate_messages
[100%] Built target msg_send_generate_messages
hscho@sparc:~/xycar_ws$
hscho@sparc:~/xycar_ws$ rosmake show msg_send/my_msg
string first_name
string last_name
int32 age
int32 score
string phone_number
int32 id_number

hscho@sparc:~/xycar_ws$ rosmake show my_msg
[msg_send/my_msg]:
string first_name
string last_name
int32 age
int32 score
string phone_number
int32 id_number

hscho@sparc:~/xycar_ws$
```



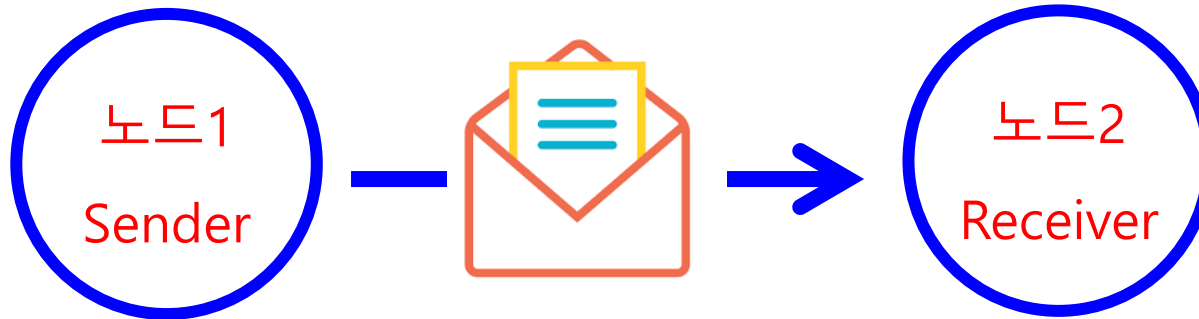
# ROS 노드 통신 프로그래밍

다양한 상황에서의 노드 통신



# 다양한 환경에서 노드간 통신이 잘 이루어지는가?

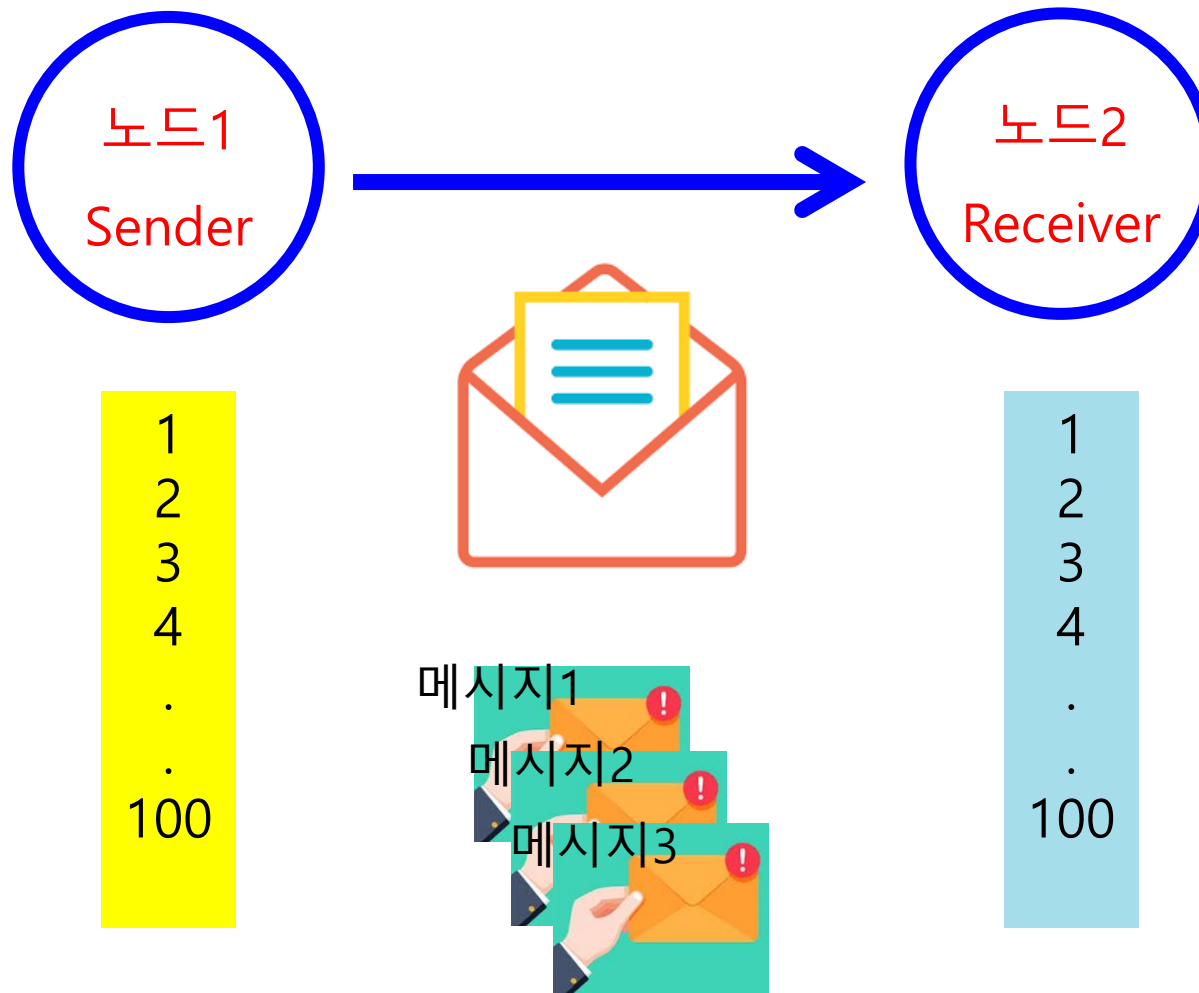
- 어떤 상황이 발생할 수 있을까? & 어떤 문제가 있을까? & 해결방법은?



- 1) 누락 없이 모두 잘 도착하는가? 특히 처음과 끝...
- 2) 데이터 크기에 따른 전송속도는 어떻게 되는가?
- 3) 도착하는 데이터를 미처 처리하지 못하면 어떻게 되는가?
- 4) 주기적 발송에서 타임슬롯을 오버하면 어떻게 되는가?
- 5) 협업해야 하는 노드를 순서대로 기동시킬 수 있는가?

# #1 궁금하다 궁금해 !?

- 누락 없이 모두 잘 도착하는가? 특히 처음과 끝...



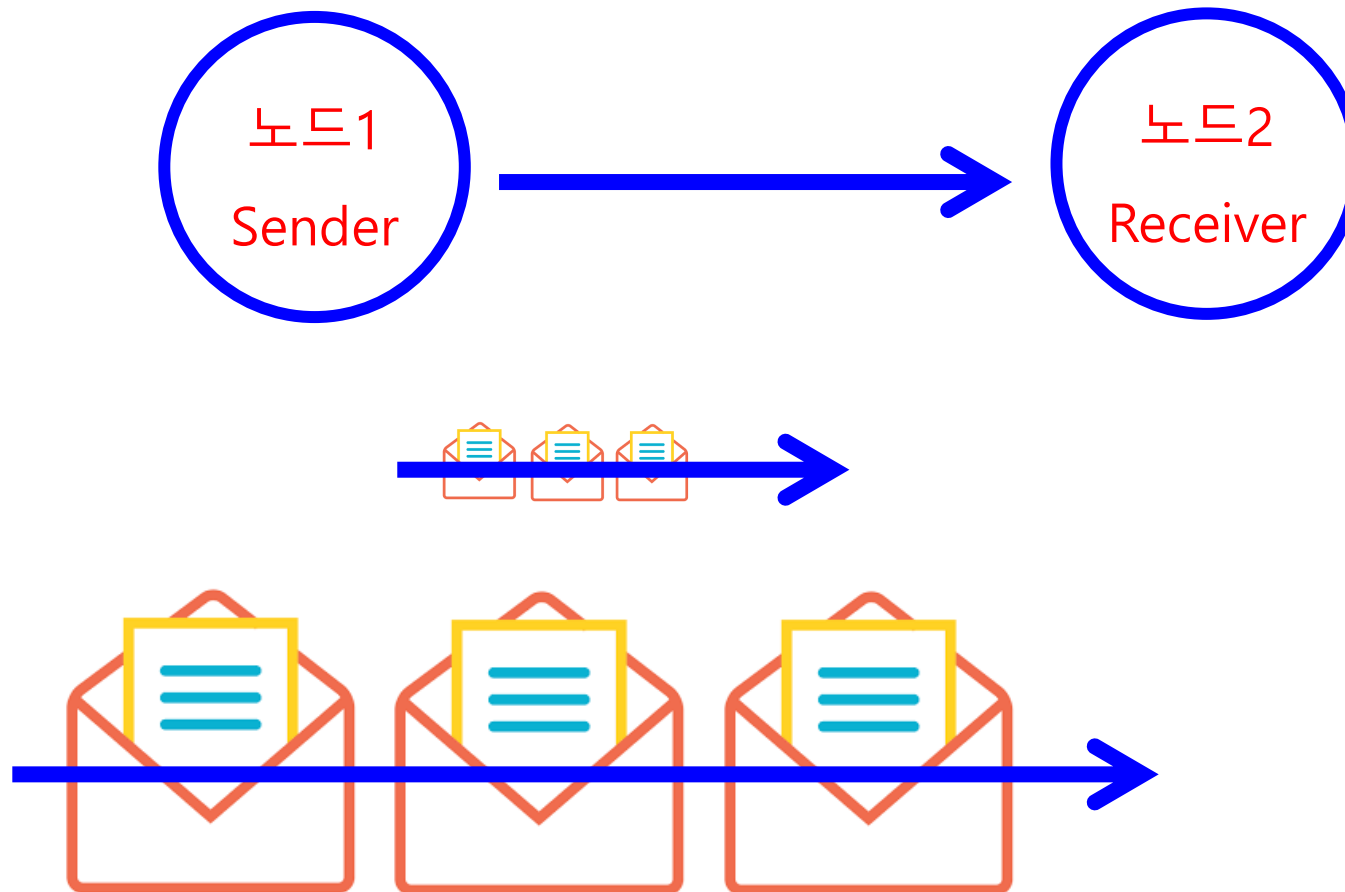
# #1 궁금하다 궁금해 !?

- 누락 없이 모두 잘 도착하는가? 특히 처음과 끝...
  - 파이썬 파일 2개랑 런치파일 1개 만들자
    - ▶ sender\_serial.py receiver\_serial.py
    - ▶ sr\_serial.launch
  - 숫자를 보내자. 그래야 받는 쪽에서 누락된 게 있는지 쉽게 알 수 있으므로...
    - ▶ 1, 2, 3, 4, ...
    - ▶ 그런데 보내는 쪽이 안 보낸 건지. 받는 쪽이 못 받은 건지 구분할 수 있을까?
  - 중간보다는 맨 처음과 끝에서 누락되는지 잘 살펴보자.
  - 받는 쪽을 먼저 실행시켜 놓고, 그 다음에 보내는 쪽을 실행시켜야 하지 않을까?
    - ▶ roslaunch 로는 노드를 순서대로 실행시킬 수 없다고 하니 rosrn 을 사용하자.
    - ▶ 더 좋고 편한 방법이 있을까?



## #2 궁금하다 궁금해 !?

- 데이터 크기에 따른 전송속도는 어떻게 되는가?

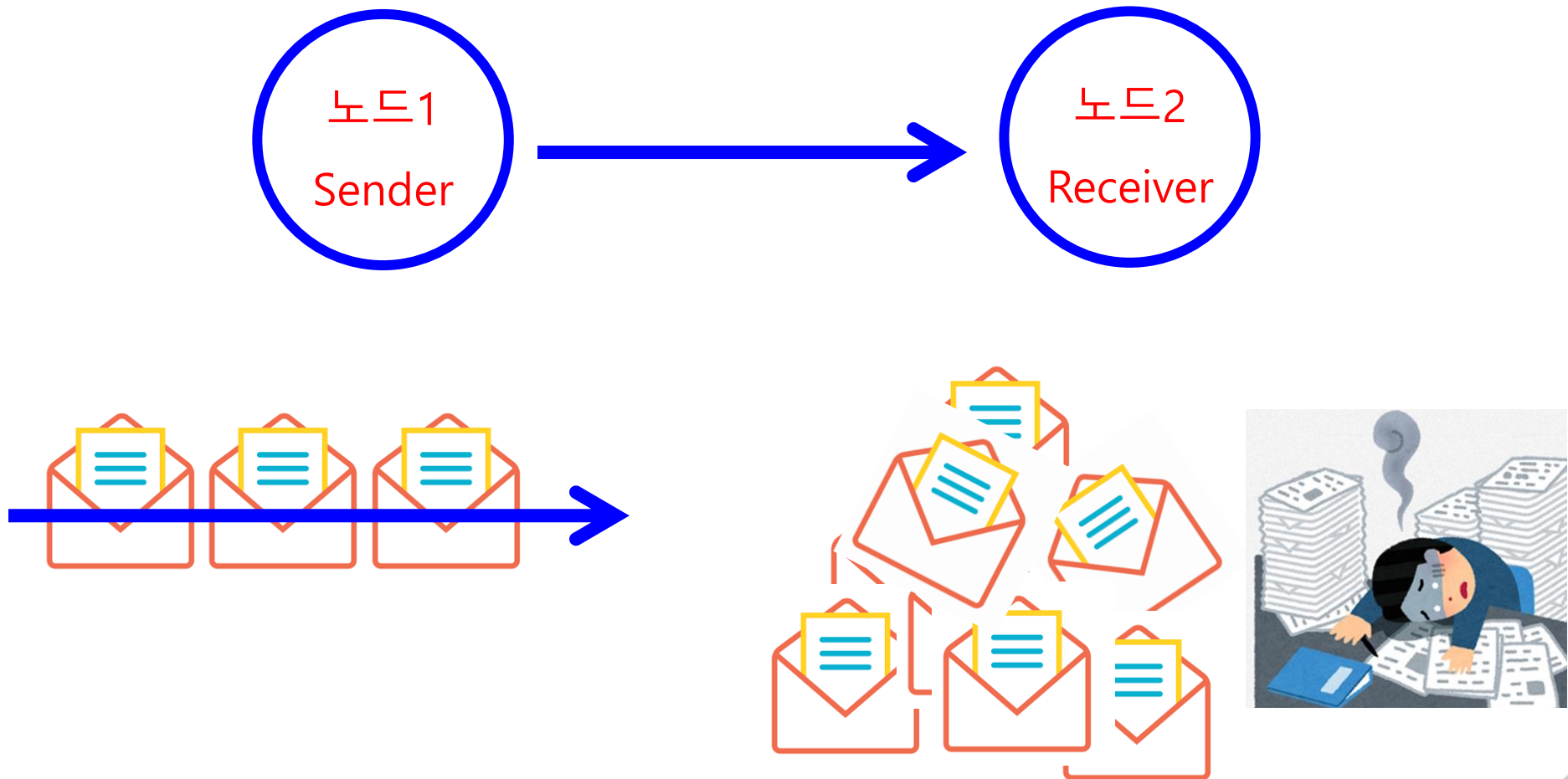


## #2 궁금하다 궁금해 !?

- 데이터 크기에 따른 전송속도는 어떻게 되는가?
  - 파이썬 파일 2개랑 런치파일 1개 만들자
    - ▶ sender\_speed.py receiver\_speed.py
    - ▶ sr\_speed.launch
  - 정해진 크기의 데이터를 반복해서 왕창 보내자.
    - ▶ 보내는 쪽은 10분 동안 시간을 정해 놓고 쉴 새 없이 보내자.
    - ▶ 10분 동안 몇 바이트 보냈는지 체크해서 송신속도 계산해 보자.
    - ▶ 받는 쪽도 10분 동안 시간을 정해 놓고, 모두 얼마나 받았는지 체크해서 수신속도를 계산해 보자.
    - ▶ 단위는 300Kbytes/sec 뭐 이렇게.
  - 받는 쪽이 없으면 어떻게 될까?
    - ▶ 토픽에 대해서 구독하는 노드가 없으면 송신속도가 더 빨라지나? 아니면 상관 없나?

### #3 궁금하다 궁금해 !?

- 도착하는 데이터를 미처 처리하지 못하면 어떻게 되는가?



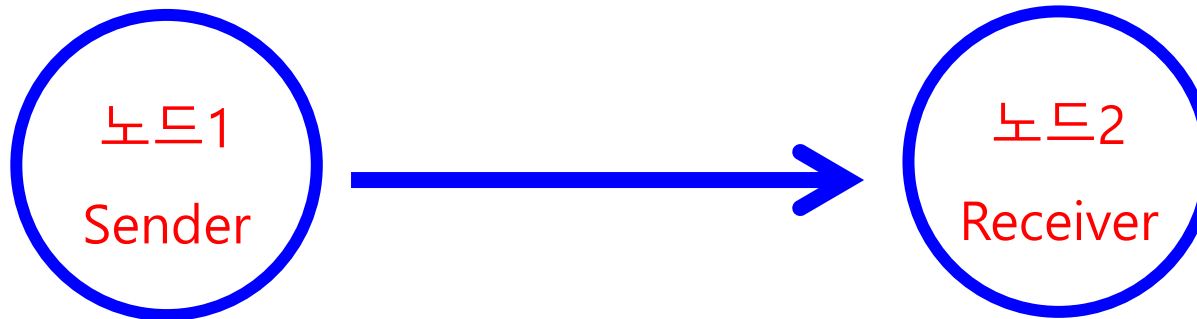
## #3 궁금하다 궁금해 !?

- 도착하는 데이터를 미처 처리하지 못하면 어떻게 되는가?
  - 파이썬 파일 2개랑 런치파일 1개 만들자
    - ▶ sender\_overflow.py receiver\_overflow.py
    - ▶ sr\_overflow.launch
  - 받는 쪽이 버벅되게 만들어 놓고 데이터를 왕창 보내자.
    - ▶ 구독자의 콜백함수 안에 시간 많이 걸리는 코드를 넣어서 토픽 처리에 시간이 걸리도록 만들자.
  - 콜백함수가 끝나지 않았는데 토픽이 새로 도착하면 어떻게 되는가?
    - ▶ 도착한 토픽은 임시로 어딘가에 쌓이는가? 그걸 나중에 꺼내서 처리할 수 있는가?
    - ▶ 아님 그냥 없어지는가? 한 번 못 받은 토픽은 영영 못 받는 것인가?
    - ▶ 발행자는 이 사실을 아는가? 알려줄 수 있는 방법이 있는가?



## #4 궁금하다 궁금해 !?

- 주기적 발송에서 타임슬롯을 오버하면 어떻게 되는가?

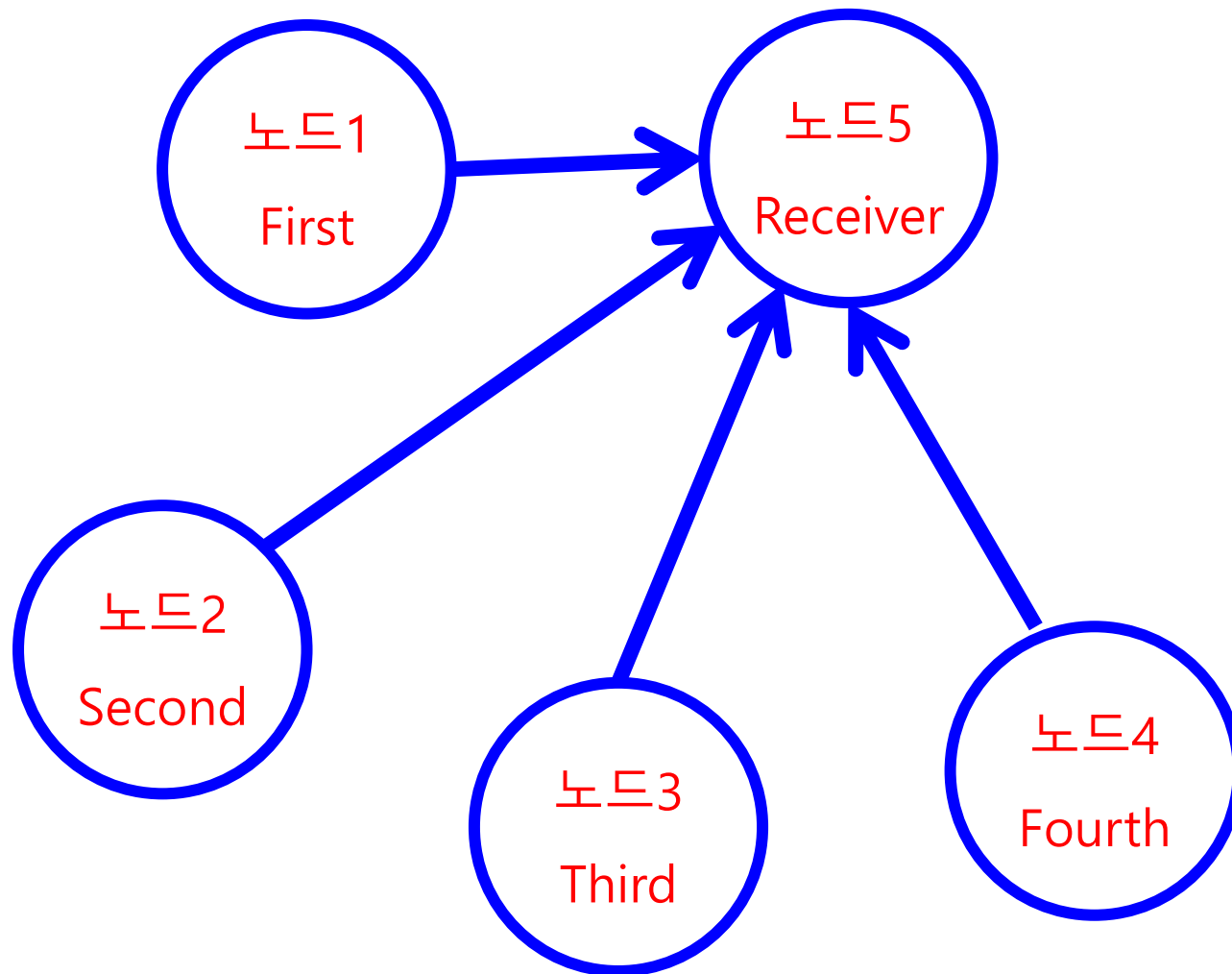


## #4 궁금하다 궁금해 !?

- 주기적 발송에서 타임슬롯을 오버하면 어떻게 되는가?
  - 파이썬 파일 2개랑 런치파일 1개 만들자
    - ▶ sender\_timeslot.py receiver\_timeslot.py
    - ▶ sr\_timeslot.launch
  - 1초에 5번 반복하게 하고 작업시간이 0.2초가 넘게 만들어 보자.
    - ▶ Rate(5) 세팅하고 sleep() 앞에 들어간 작업코드에 대해서 수행시간을 늘려보자.
    - ▶ 늘렸다 줄였다 변동성 있게 해보자. 입력값을 받아서 이걸 조정할 수 있게 만들까?
  - 1초에 5번 규칙을 지킬 수 없으면 어떻게 할까?
    - ▶ 앞에서부터 쪽 밀리는 식으로 일할까?
    - ▶ 쉬는 시간을 조정할까?
    - ▶ 이번엔 3번만 하고 다음번을 기약할까?

## #5 궁금하다 궁금해 !?

- 협업해야 하는 노드를 순서대로 기동시킬 수 있는가?

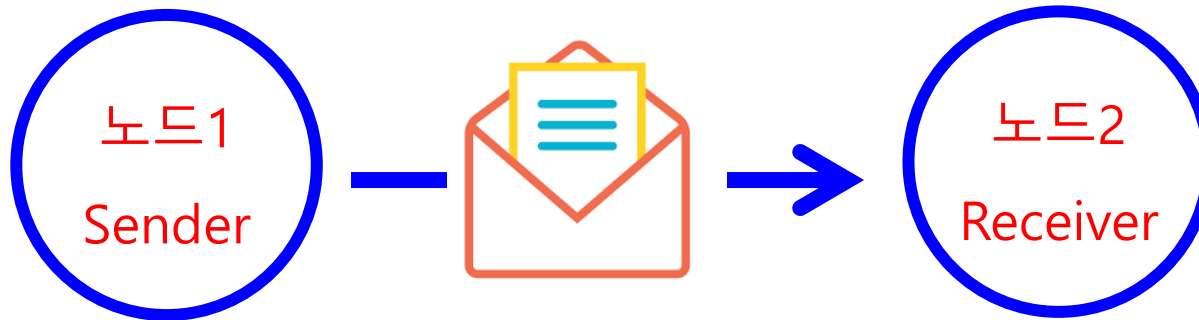


## #5 궁금하다 궁금해 !?

- 협업해야 하는 노드를 순서대로 기동시킬 수 있는가?
- 파이썬 파일 5개랑 런치파일 1개 만들자
  - ▶ `first.py second.py third.py fourth.py receiver.py`
  - ▶ `sr_order.launch`
- 순서대로 receiver에 메시지를 보내도록 만들자.
  - ▶ receiver는 도착한 순서대로 출력한다. First – Second – Third – Fourth 이렇게 되어야 한다.
  - ▶ 앞에 있는 노드가 움직이기 전에 먼저 움직여서는 안된다. (움직인다 = 토픽을 보내는 걸로 대신하자)
- 어떻게 동기를 맞추고 순서를 맞출 수 있을까?
  - ▶ Launch 파일을 이용해서 할 수 있을까?
  - ▶ ROS 의 도움으로 할 수 있을까?
  - ▶ 아니면 내가 프로그래밍 해서 해야 하는가? 한번 해볼까?

# 다양한 환경에서 노드간 통신이 잘 이루어지는가?

- 여러분들도 곰곰히 생각해 보고 고민해 보세요...



- 1) 누락 없이 모두 잘 도착하는가? 특히 처음과 끝...
- 2) 데이터 크기에 따른 전송속도는 어떻게 되는가?
- 3) 도착하는 데이터를 미처 처리하지 못하면 어떻게 되는가?
- 4) 주기적 발송에서 타임슬롯을 오버하면 어떻게 되는가?
- 5) 협업해야 하는 노드를 순서대로 기동시킬 수 있는가?



# Q&A

---

감사합니다.



(주)자이트론

허성민

smher@xytron.co.kr

