ROS 단어 설명

ROS(Robot OS)

로스는 로봇 응용 프로그램을 개발을 위한 운영체제와 같은 로봇 플랫폼이다. 로스는 로봇 응용프로그램을 개발할때 필요한 하드웨어 추상화, 하위 디바이스 제어, 일반적으로 사용되는 기능의 구현, 프로세스간의 메시지 파싱, 패키지 관리, 개발환경에 필요한 라이브러리와 다양한 개발 및 디버깅 도구를 제공한다

마스터(master)

마스터는 노드와 노드사이의 연결 및 메시지 통신을 위한 네임 서버와 같은 역할을 한다. 로스코어(roscore)가 실행 명령어 이며, 마스터를 실행하게되면 각 노드들의 이름을 등록하고 필요에 따라 정보를 받을 수 있다. 마스터가 없이는 노드간의 접속, 토픽과 서비스와 같은 메시지 통신을 할 수 없다.

마스터는 마스터에 접속하는 슬레이브들과의 접속 상태를 유지하지 않는 HTTP 기반의 프로토콜인 XMLRPC를 이용하여 슬레이브들과 통신하게 된다. 즉, 슬레이브인 노드들이 필요할때만 접속하여 자신의 정보를 등록하거나 다른 노드의 정보를 요청하여 수신받을 수 있다. 평상시에는 서로간의 접속 상태를 체크하지 않는다. 이러한 기능으로 매우 크고, 복잡한 환경에서도 적용가능하다. 또한, XMLRPC는 매우 가볍고, 다양한 프로그래밍 언어를 지원하고 있기때문에 다기종 하드웨어, 다언어를 지원하는 로스에 매우 적합하다.

로스를 구동하게 되면 사용자가 정해놓은 ROS_MASTER_URI 변수에 기재되어 있는 URI 주소 및 포트를 갖는다. 사용자가 설정해놓지 않은 경우에는 URI 주소로 현재의 로컬 IP 를 사용하고, 11311 포트를 이용하게 된다.

노드(Node)

로스에서 최소 단위의 실행 프로세서를 가르키는 용어이다. 하나의 실행 가능한 프로그램으로 생각하면 된다.

로스에서는 하나의 목적에 하나의 노드를 작성하길 권하고 있으며 재사용이 쉽도록 구성하여 만들도록 권고하고 있다. 예를들어, 모바일 로봇의 경우, 로봇을 구동하기 위하여 각 프로

그램을 세분화 시킨다. 예를들어, 센서 드라이브, 센서 데이타를 이용한 변환, 장애물 판단, 모터 구동, 엔코더 입력, 네이게이션 등 세부화된 작은 노드들을 이용한다.

노드는 생성과 함께 마스터에 노드이름, 발행자이름, 구독자이름, 토픽이름, 서비스이름, 메시지형태, URI 주소 및 포트를 등록한다. 이 정보들을 기반으로 각 노드는 노드끼리 토픽 및 서비스를 이용하여 메시지를 주고 받을 수 있다.

노드는 마스터와 통신할 때, XMLRPC를 이용하며, 노드간의 통신에서는 XMLRPC 및 TCP/IP 통신 계열의 TCPROS를 이용하고 있다. 노드간의 접속 요청 및 응답은 XMLRPC를 사용하며, 메시지 통신은 마스터와는 관계없이 노드와 노드간의 직접적인 통신으로 TCPROS를 이용하고 있다. URI 주소 및 포트는 현재 노드가 실행중인 컴퓨터에 저장된 ROS_HOSTNAME 라는 환경 변수값을 URI 주소로 사용하며, 포트는 임의적 고유의 값으로 설정되게 된다.

패키지(package)

로스를 구성하는 기본 단위로써 실행 가능한 노드를 포함하고 있다. 로스는 패키지를 단위로 각각의 응용 프로그램들이 개발된다. 패키지는 최소한 하나 이상의 노드를 포함하고 있다. ROS Hydro의 경우 공식적으로 약 700개 의 패키지를 제공하고 있으며, 유저들이 개발하여 공개된 패키지가 대략 3300개에 달하고 있다.

메타패키지

(**metapackage**)

공통된 목적을 가지는 패키지들을 모아둔 패키지들의 집합을 말한다. 복수의 패키지를 포함하고 있다

메시지

(message,msg)

노드는 메시지를 통해 노드간의 데이터를 주고받게 된다. 메시지는 integer, floating point, boolean 와 같은 변수형태이다. 또한, 메시지안에 메시지를 품고 있는 간단한 데이터 구조 및 메시지들의 배열과 같은 구조도 사용할 수 있다.

메세지를 이용한 통신방법으로는 TCPROS, UDPROS 방식등이 있으며, 단방향 메시지 송/수신 방식의 토픽과 양방향 메시지 요청/응답 방식의 서비스를 이용하고 있다.

토픽(topic)

토픽은 "이야깃거리"이다. 발행자 노드가 하나의 이야깃거리에서 대해서 토픽이라는 이름으로 마스터에 등록한 후, 이야깃거리에 대한 이야기를 메시지 형태로 발행한다. 이 이야깃거리를 수신 받기를 원하는 구독자 노드는 마스터에 등록된 토픽의 이름에 해당되는 발행자 노드의 정보를 받는다. 이 정보를 기반으로 구독자 노드는 발행자 노드와 직접적으로 연결하여 메시지를 송/수신 또는 요청/응답 받게 된다.

발행(publish)

및 발행자(Publisher)

발행은 토픽의 내용에 해당되는 메시지 형태의 데이터를 송신하는 것을 말한다.

발행자 노드는 발행을 수행하기 위하여 토픽을 포함한 자신의 정보들을 마스터에 등록하고, 구독을 원하는 구독자 노드에게 메시지를 보내게 된다. 발행자는 이를 실행하는 개체로써 노드에서 선언하게 된다. 발행자는 하나의 노드에서 복수로 선언이 가능하다.

구독(subscribe)

및 구독자(Subscriber)

구독은 토픽의 내용에 해당되는 메시지 형태의 데이터를 수신하는 것을 말한다.

구독자 노드는 구독을 수행하기 위하여 토픽을 포함한 자신의 정보들을 마스터에 등록하고, 구독하고자 하는 토픽을 발행하는 발행자 노드의 정보를 마스터로부터 받는다. 이 정보를 기반으로 구독자 노드는 발행자 노드와 직접적으로 접속하여 발행자 노드로부터 메시지를 받게 된다. 구독자는 이를 실행하는 개체로써 노드에서 선언하게 된다. 구독자는 하나의 노드에서 복수로 선언이 가능하다

서비스(service)

발행과 구독 개념의 토픽 통신 방식은 비동기 방식이라 필요에 따라서 주어진 데이터를 전송하고 받기에 매우 훌륭한 방법이다. 또한, 한번의 접속으로 지속적인 메시지를 송/수신하기 때문에 지속적으로 메시지를 발송해야하는 센서 데이터에 적합하여 많이 사용되고 있다.

하지만, 경우에 따라서는 요청과 응답이 함께 사용되는 동기 방식의 메시지 교환 방식도 필요하다. 이에따라, 로스에서는 서비스라는 이름으로 메시지 동기 방식을 제공하고 있다.

서비스는 요청이 있을 경우에 응답을 하는 서비스 서버와 요청을 하고 응답을 받는 서비스 클라이언트로 나뉘어 있다. 서비스는 토픽과는 달리 1회성 메시지 통신이다. 서비스의 요청과 응답이 완료되면 연결된 두 노드의 접속은 끊기게 된다

서비스 서버

(service server)

서비스 서버는 요청을 입력으로 받고, 응답을 출력으로 하는 서비스 메시지 통신의 서버 역할을 말한다. 요청과 응답은 모두 메시지로 되어 있으며, 서비스 요청에 의하여 주어진 서비스를 수행 후에 그 결과를 서비스 클라이언트에게 전달한다. 서비스 메시지 방식은 동기식이기에 정해진 명령을 지시 및 수행하는 노드에 사용하는 경우가 많다.

서비스 클라이언트

(service client)

서비스 클라이언트는 요청을 출력으로 하고, 응답을 입력으로 받는 서비스 메시지 통신의 클라이언트 역할을 말한다. 요청과 응답은 모두 메시지로 되어 있으며, 서비스 요청을 서비스 서버에 전달하고 그 결과 값을 서비스 서버로 부터 받는다. 서비스 메시지 방식은 동기식이기에 정해진 명령을 지시 및 수행하는 노드에 사용하는 경우가 많다

캐킨(catkin)

로스의 빌드 시스템을 말한다. 로스의 빌드 시스템은 기본적으로 CMake(Cross Platform Make) 를 이용하고 있어서 패키지 폴더에 CMakeLists.txt 라는 파일에 빌드 환경을 기술하고 있다. 로스에서는 CMake 를 로스에 맞도록 수정하여 로스에 특화된 캐킨 빌드 시스템을 만들었다. 캐킨 빌드 시스템은 로스와 관련된 빌드, 패키지 관리, 패키지간의 의존관계 등을 편리하게 사용할 수 있도록 하고 있다.

로스빌드(rosbuild)

catkin 빌드 시스템의 이전에 사용되었던 빌드 시스템이다. ROS Fuerte 버전부터 rosbuild 빌드 시스템 대신에 catkin 빌드 시스템을 사용하고 있으나, rosbuild 빌드 시스템 또한 사용가능하다. 하지만, 이는 ROS 버전의 호환성을 위해 남겨둔 것이지 공식적으로는 추천하지 않는다. 만약, rosbuild 빌드 시스템을 사용한 이전 패키지를 사용해야만 한다면 rosbuild를 catkin으로 변경하여 사용하기를 추천한다

로스코어(roscore)

로스 마스터를 구동하는 명령어이다. 같은 네트웍이라면 다른 컴퓨터에서 실행하여도 된다. 로스를 구동하게 되면 사용자가 정해놓은 ROS_MASTER_URI 변수에 기재되어 있는 URI 주소 및 포트를 갖는다. 사용자가 설정해놓지 않은 경우에는 URI 주소로 현재의 로컬 IP 를 사용하고, 11311 포트를 이용하게 된다.

매개변수(parameter)

노드에서 사용되는 매개변수를 말한다. 흔히, 윈도우즈 프로그램에서 *.ini 설정파일과 같다고 생각하면 된다. 디폴트로 설정 값들이 지정되어 있고, 필요에 의해서 외부에서 이 매개변수를 읽기, 쓰기가 가능하다. 특히, 상황에 맞추어 이 매개변수를 외부에서 쓰기기능을 이용하여 설정값을 실시간으로 바꿀수 있기에 매우 유용한 방법이다. 예를들어 접속하는 USB포트 및 카메라 캘리브레이션 값, 속도 및 명령어들의 최대/최저 값 등의 설정등을 지정할 수 있다.

매개변수 서버

(parameter server)

매개변수 서버는 패키지에서 매개변수를 사용할 때, 각 매개변수를 등록하는 서버를 말한다. 매개변수 서버는 마스터의 일부분이다.

로스런(rosrun)

로스의 기본적인 실행 명령어이다. 패키지에서 하나의 노드를 실행하는데 사용된다. 노드가 사용하는 URI 주소 및 포트는 현재 노드가 실행중인 컴퓨터에 저장된 ROS_HOSTNAME 라는 환경 변수값을 URI 주소로 사용하며, 포트는 임의적 고유의 값으로 설정되게 된다.

로스런치(roslaunch)

로스런(rosrun)이 하나의 노드를 실행하는 명령어라면 로스런치(roslaunch)는 복 수개의 노드를 실행하는 개념이다. 이 명령어를 통해 정해진 단일 혹은 복수의 노드를 실행시킬 수 있다.

그 이외의 기능으로 실행시에 패키지의 매개변수를 변경, 노드 명의 변경, 노드 네임 스페이스 설정, ROS_ROOT 및 ROS_PACKAGE_PATH 설정, 이름 변경, 환경 변수 변경 등의 실행시 변경할 수 있는 많은 옵션들을 갖춘 노드 실행에 특화된 로스 명령어이다.

로스런치는 "*.launch" 라는 로스런치파일을 사용하여 실행 노드에 대한 설정을 해주는데 이는 XML 기반으로 되어 있으며, 태그별 옵션을 제공하고 있다. 실행 명령어로는 "roslaunch 패키지명 로스런치파일" 이다.

배그(bag)

로스에서 주고받는 메시지의 데이터를 저장할 수 있는 있는데 이를 배그라고 한다. ROS에서는 이 배그를 이용하여 메시지를 저장하고 필요로 할 때 이를 재생하여 이전 상황을 그대로 재현할 수있는 기능을 갖추고 있다. 예를들어, 센서를 이용한 로봇 실험을 실행할 때, 센

서 값을 배그를 이용하여 메시지 형태로 저장한다. 이 저장된 메시지는 같은 실험을 수행하지 않아도 저장해둔 배그 파일을 재생하는 것으로 그 당시의 센서값을 반복 사용가능하다. 특히, 기록, 재생의 기능을 활용하여 반복되는 프로그램 수정이 많은 알고리즘 개발에 매우 유용하다.

로스위키(ros wiki)

로스의 각 패키지 및 기능들을 설명하는 페이지(http://wiki.ros.org/)이다. 각 패키지는 위키 패이지를 가지고 있어서 패키지에 대한 간단한 설명, 사용되는 매개변수, 저작자, 라이선스, 홈페이지, 저장소, 튜토리얼등을 포함하고 있다.

저장소(repositoy)

공개된 패키지의 경우, 각 패키지의 위키에 저장소를 명시하고 있다. 저장소는 패키지가 저장된 웹상의 저장소 URL 주소이며 svn, hg, git 등의 소스 관리 시스템을 이용하여 이슈, 개발, 다운로드 등을 관리하고 있다.

그래프(graph)

위에서 설명한 노드, 토픽, 발행자, 구독자 관계를 그래프를 통해 나타나게 하는 것이다. 현재 실행중인 메시지 통신을 그래프화 시킨 것으로 1회성인 서비스에 대한 그래프는 작성할 수 없다. 실행은 rqt_graph 패키지의 rqt_graph 노드를 실행하면 된다. "rqt_graph" 또는 "rosrun rqt_graph rqt_graph" 명령어를 이용하면 된다.

이름(name)

노드, 매개변수, 토픽, 서비스는 모두 이름을 갖고 있다. 이 이름은 마스터에 등록하고 각 노드의 매개변수, 토픽, 서비스를 사용할때 이 이름을 기반으로 상호적으로 동작하도록 되어있다. 또한, 이름은 실행시에 변경가능하기 때문에 매우 유연하고, 같은 노드, 매개변수, 토픽, 서비스라고 하여도 다른 이름으로 중복 실행이 가능하다. 이러한 이름의 사용으로 로스는 큰 규모의 프로젝트, 복잡한 구조의 시스템에도 적합하다.

클라이언트 라이브러리

(client libray)

로스는 사용되는 언어의 의존성을 낮추기 위하여 클라이언트 라이브러리라는 이름으로 각종 언어의 개발환경을 제공하고 있다.

주요한 클라이언트 라이브러리로는 C++, Python, Lisp 등이 있으며, 그 이외에도 java, lua, .NET, EusLisp, R 등의 언어들을 사용가능하다. 이를 위해 roscpp, rospy, roslisp, rosjava, roslua, roscs, roseus, PhaROS, rosR 등의 클라이언트 라이브러리가 개발되었다

URI

(Uniform Resource Identifier)

URI (Uniform Resource Identifier, 통합 자원 식별자) 는 인터넷에 있는 자원을 나타내는 유일한 주소이다. URI 은 인터넷에서 요구되는 기본조건으로서 인터넷 프로토콜에서 식별자로 사용된다.

MD5

(Message-Digest algorithm 5)

MD5는 128비트 암호화 해시 함수이다. 주로 프로그램이나 파일이 원본 그대로인지를 확인하는 무결성 검사 등에 사용된다. 로스에서의 메시지를 이용한 통신에서 MD5를 이용하여 메시지 송수신의 무결성 검사를 하고 있다.

RPC

(Remote Procedure Call)

RPC란 '멀리 떨어져(Remote) 있는 컴퓨터상의 프로그램이 다른 컴퓨터 내에 있는 서브프로그램(Procedure)을 불러내는(Call)' 것을 의미한다. 컴퓨터 프로그램이 다른 주소 공간에서 원격 제어를 위한 프로그래머의 세세한 코딩 없이 함수나 프로시저의 실행을 허용하는 기술로써 대표적으로는 TCP/IP, IPX 등의 전송 프로토콜을 이용한다.

XML

(Extensible Markup Language)

XML은 W3C에서 다른 특수 목적의 마크업 언어를 만드는 용도에서 권장되는 다목적 마크업 언어(markup language)로 태그 등을 이용하여 데이터의 구조를 명기하는 언어의 한 가지이다.

XMLRPC

XML-RPC란, RPC 프로토콜의 일종으로서, 인코딩 형식에서는 XML을 채택하고, 전송 방식에서는 접속 상태를 유지하지 않고 체크하지 않는 요청/응답 방식의 HTTP 프로토콜을 사용하고 있다. XML-RPC는 매우 단순한 규약으로서, 작은 데이터 형식이나 명령을 정의하는 정도로만 사용하고 있어서 꽤나 단순한 편이다. 이러한 특징으로, XMLRPC는 매우 가볍고, 다양한 프로그래밍 언어를 지원하고 있기 때문에 다기종 하드웨어, 다언어를 지원하는 로스에 매우 적합하다.

TCP/IP

(Transmission Control Protocol / Internet Protocol)

TCP 는 Transmission Control Protocol 의 약자로써, 전송 제어 프로토콜이라고 부른다. 흔히 TCP/IP 라 부르는데 이는 인터넷 프로토콜 계층의 시각에서 보면 IP(Internet Protocol) 를 기반으로 전송 제어 프로토콜인 TCP 를 사용하여 데이터의 전달을 보증하고 보낸 순서대로 송/수신하게 된다.

TCPROS

메시지 및 서비스에서 사용되는 TCP/IP 기반의 메시지 방식을 TCPROS라고 한다.

UDPROS

메시지 및 서비스에서 사용되는 UDP 기반의 메시지 방식을 UDPROS라고 한다. 일반적으로 ROS에서는 사용되지 않고 있다

CMakeLists.txt

로스의 빌드 시스템인 캐킨은 기본적으로 CMake를 이용하고 있어서 패키지 폴더에 CMakeLists.txt 라는 파일에 빌드 환경을 기술하고 있다.

package.xml

패키지의 정보를 담은 XML 파일로써 패키지의 이름, 저작자, 라이선스, 의존성 패키지 등을 기술하고 있다.)