# ActiveMQ--1.客户端与服务器之间的传输连接

ActiveMQ的cilent-to-broker的连接，叫做传输连接(Transport connectors)，broker-to-broker间的连接，叫做网络连接(Network connectors)。

ActiveMQ支持许多种客户端与服务器的传输连接。分别是TCP，NIO，UDP，SSL，HTTP(S)，VM，AMQP，MQTT，Peer，Multicast，WebSockets。

使用方法是指定连接Broker URI。URI scheme是传输连接名称，path是Broker的IP地址和端口号，另外可以使用Key Value形式的Query串作为参数，如：tcp://localhost:61616?trace=true。而且各个协议之间可以组合，使用方式如：amqp+ssl://localhost:5671。

为ActiveMQ服务器配置多协议支持的方法是，编辑%ACTIVEMQ\_HOME%conf\activemq.xml文件。此文件是ActiveMQ的配置文件。编辑<transportConnectors>节点，配置片段如下：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | < transportConnectors >       < transportConnector  name = "openwire"  uri = "<tcp://localhost:61616>"  />       < transportConnector  name = "ssl"  uri = "<ssl://localhost:61617>" />       < transportConnector  name = "stomp"  uri = "<stomp://localhost:61613>" />       < transportConnector  name = "ws"  uri = "<ws://localhost:61614/>"  />       < transportConnector  name = "amqp+ssl"  uri = "amqp+<ssl://localhost:5671/>"  />  </ transportConnectors > |

其中name和uri必须唯一。这样ActiveMQ的Broker就开启的不同的监听端口处理不同的传输连接。不同的传输连接的优缺点及使用场景如下：

**TCP：**ActiveMQ默认的传输连接，也是最常用的使用方式。长连接，每个客户端实例都会与服务器维持一个连接。每个连接一个线程。TCP的优点是：

性能高：ActiveMQ使用默认协议OpenWire序列化和反序列化消息。OpenWire是一个性能很高的序列化协议。

可用性高：TCP是使用最广泛的技术，几乎所有的开发语言都支持TCP协议。

可靠性高：TCP协议确保消息不会在网络传说的过程中丢失。

tcp配置：

|  |  |
| --- | --- |
| 1 | < transportConnector  name = "tcp"  uri = "<tcp://localhost:61616>" /> |

**NIO：**使用Java的NIO方式对连接进行改进，因为NIO使用线程池，可以复用线程，所以可以用更少的线程维持更多的连接。如果有大量的客户端，或者性能瓶颈在网络传输上，可以考虑使用NIO的连接方式。也可以根据不同的场景选择不用的传输连接，比如：Producer有很多，但是Consumer很少，可以Producer用NIO协议，Consumer用TCP协议。从ActiveMQ 5.6版本开始，NIO可以支持和SSL搭配使用的传输连接。

nio配置：

|  |  |
| --- | --- |
| 1 | < transportConnector  name = "nio"  uri = "<nio://localhost:61616>" /> |

nio+ssl配置：

|  |  |
| --- | --- |
| 1 | < transportConnector  name = "nio+ssl"  uri = "nio+<ssl://localhost:61616>" /> |

**UDP：**与面向连接，可靠的字节流服务的TCP不同，UDP是一个面向数据的简单传输连接，没有TCP的三次握手，所以性能大大强于TCP，但是是以牺牲可靠性为前提。适用于丢失也无所谓的消息，如统计uv，pv。（当然如果真是统计uv什么的，有Kafka这样专门的消息中间件）。

udp配置：

|  |  |
| --- | --- |
| 1 | < transportConnector  name = "udp"  uri = "<udp://localhost:8123>" /> |

**SSL：**需要一个安全连接的时候可以考虑使用SSL，适用于client和broker在公网的情况，如使用aws云平台等。

ssl配置：

|  |  |
| --- | --- |
| 1 | < transportConnector  name = "ssl"  uri = "<ssl://localhost:8123>" /> |

**HTTP(S)：**需要穿越防火墙，可以考虑使用HTTP(S)，但由于HTTP(S)是短连接，每次创建连接的成本较高，所以性能最差。

http配置：

|  |  |
| --- | --- |
| 1 | < transportConnector  name = "http"  uri = "<http://localhost:8080>" /> |

https配置：

|  |  |
| --- | --- |
| 1 | < transportConnector  name = "https"  uri = "<https://localhost:8080>" /> |

**VM：**勉强可以算一个协议吧。使用场景是client和broker在同一个Java虚拟机的情况。如使用代码启动嵌入式的ActiveMQ Broker实例，通常用于单元测试。因为是嵌入式，所以不需要配置ActiveMQ的配置文件，只要在连接Broker的URI种直接使用即可。vm配置：(spring配置片段)

|  |  |
| --- | --- |
| 1  2  3 | < bean  id = "connectionFactory"  class = "org.apache.activemq.ActiveMQConnectionFactory"  depends-on = "broker" >       < property  name = "brokerURL"  value = "<vm://localhost>" />  </ bean > |

**AMQP：** ActiveMQ 5.8新增加的传输连接。用于支持AMQP（高级消息队列协议）。因为AMQP是消息队列的标准协议，而且已经越来越被广泛使用，所以ActiveMQ也支持了此协议。AMQP协议可以搭配NIO或SSL协议使用，amqp+nio用于提升系统的延展性和性能。amqp+ssl可以创建安全连接。

amqp配置：

|  |  |
| --- | --- |
| 1 | < transportConnector  name = "amqp"  uri = "<amqp://localhost:5672>" /> |

amqp+nio配置：

|  |  |
| --- | --- |
| 1 | < transportConnector  name = "amqp+nio"  uri = "amqp+<nio://localhost:5672>" /> |

amqp+ssl配置：

|  |  |
| --- | --- |
| 1 | < transportConnector  name = "amqp+ssl"  uri = "amqp+<ssl://localhost:5672>" /> |

**MQTT：**ActiveMQ 5.8新增加的传输连接。是一个轻量级的消息订阅/发布协议。和AMQP一样，同样支持搭配NIO或SSL使用。

mqtt配置：

|  |  |
| --- | --- |
| 1 | < transportConnector  name = "mqtt"  uri = "<mqtt://localhost:1883>" /> |

mqtt+nio配置：

|  |  |
| --- | --- |
| 1 | < transportConnector  name = "mqtt+nio"  uri = "mqtt+<nio://localhost:1883>" /> |

mqtt+ssl配置：

|  |  |
| --- | --- |
| 1 | < transportConnector  name = "mqtt+ssl"  uri = "mqtt+<ssl://localhost:1883>" /> |