

ActiveMQ

IT培优·李康
2017-5-8

教学大纲

- ◆ActiveMQ介绍
- ◆JMS简介
- ◆ActiveMQ主要特性
- ◆ActiveMQ功能
- ◆入门实例
- ◆集成Spring
- ◆常见协议

ActiveMQ介绍

- ActiveMQ是什么？

- ActiveMQ 是Apache出品，最流行的，能力强劲的开源消息总线。ActiveMQ 是一个完全支持JMS1.1和J2EE 1.4规范的 JMS Provider实现。

JMS 简介

- JMS (Java Message Service) , 即 : java消息服务应用程序接口。
- 是Java平台面向消息中间件 (MOM) 的API/技术规范。
- 场景 : 应用与两个应用程序之间 , 或者分布式系统架构中分发消息 , 可进行异步/同步方式的通讯 , 和平台API无关 , 基本多数的MOM都提供对JMS的支持。

JMS 体系架构

- JMS提供者
 - 连接面向消息中间件的，JMS接口的一个实现。提供者可以是Java平台的JMS实现，也可以是非Java平台的面向消息中间件的适配器。
- JMS客户
 - 生产或消费基于消息的Java的应用程序或对象。
- JMS生产者
 - 创建并发送消息的JMS客户。
- JMS消费者
 - 接收消息的JMS客户。
- JMS消息
 - 包括可以在JMS客户之间传递的数据的对象。
- JMS队列
 - 一个容纳那些被发送的等待阅读的消息的区域。与队列名字所暗示的意思不同，消息的接受顺序并不一定要与消息的发送顺序相同。一旦一个消息被阅读，该消息将被从队列中移走。
- JMS主题
 - 一种支持发送消息给多个订阅者的机制。

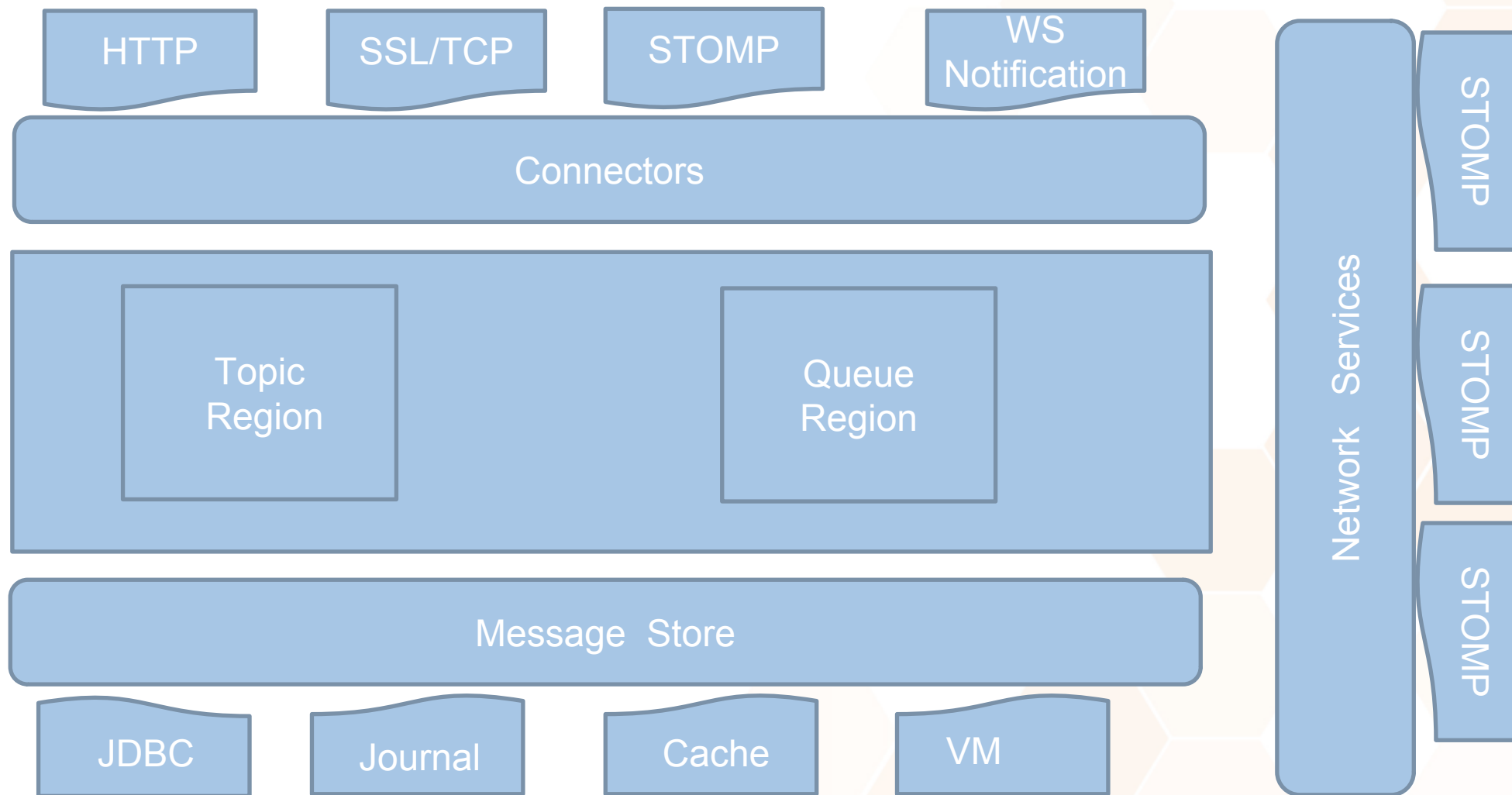
JMS 属性

- Destination (接口/目标)
- Product (生产者)
- Consumer (消费者)
- Broker (消息转发器)
- P2P, Pub/Sub (模型)
 - P2P:消息队列(Queue)、发送者(Sender)、接收者(Receiver)
 - Pub/Sub:主题 (Topic)、发布者 (Publisher)、订阅者 (Subscriber)

ActiveMQ与JMS

- JMS是一种规范
- ActiveMQ是JMS规范的一种实现

AvtiveMQ架构图



ActiveMQ主要特性

- (1)JMS1.1、J2EE1.4
- (2)J2EE servers(Tomcat,JBoss4,GlassFish,WebLogic...)
- (3)多语言客户端 (Java,C,C++,C#,Ruby,PhP)
- (4)多种协议(VM,TCP,SSL,UDP,multicast,JGroups...)
- (5)Spring
- (6)Ajax
- (7)CXF,Axis (WebService的两个流行的框架)
- (8)REST (状态传递)
- (9)Message Groups,Virtual Destinations,Wildcards,Composite , Destinations
- (10)持久化(journal,JDBC)
- (11)性能(client-server,cluster,peer...)

ActiveMQ功能

- 多种协议
- 持久化
- 安全
- 群集
- 监控
- 其他

AvtiveMQ多种协议

URI: *scheme:scheme-specific-part*

- VM vm://brokername
- TCP tcp://host:port
- SSL ssl://host:port
- HTTP http://host:port
- UDP udp://host:port
- peer peer://group/brokername
- multicast multicast://IPAddress
- static static(list uris)
- failover failvoer(list uris)
- discovery discovery://host:port

ActiveMQ持久化

- **日志**

<journaledJDBC journalLogFiles="5" dataDirectory="../mq-data" />

- **数据库**

包括：**Derby, HSQL, MySQL, SQLServer, Sybase, DB2, Oracle...**

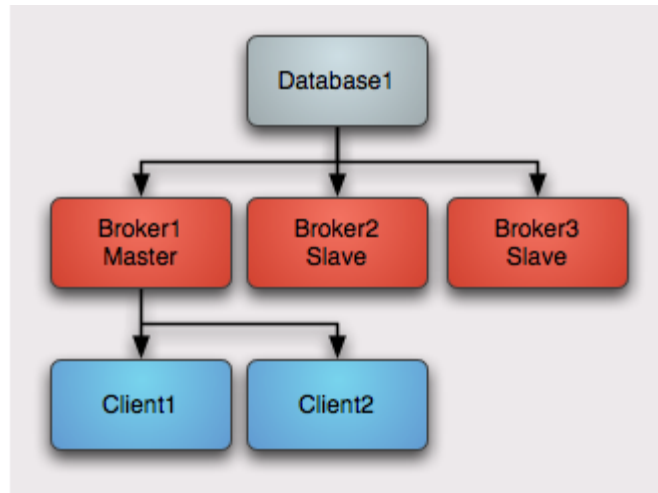
<journaledJDBC dataSource="#mysql-ds"/>

ActiveMQ安全

- **认证**
simpleAuthenticationPlugin
jaasAuthenticationPlugin
- **授权**
authorizationPlugin

ActiveMQ集群

- Master/Slave
- Network of Brokers



Active监控

- **JMX(Java管理扩展框架)**
- **Advisory Message (通知消息)**

Queue与Topic区别

类型	Topic	Queue
概要	发布/订阅消息	点对点
状态	topic数据默认不落地，是无状态的	Queue数据默认会在mq服务器上以文件形式保存，比如Active MQ一般保存在\$AMQ_HOME\data\kr-store\data下面。也可以配置成DB存储
完整性保障	并不保证publisher发布的每条数据，Subscriber都能接受到	Queue保证每条数据都能被receiver接收
消息是否会丢失	一般来说publisher发布消息到某一个topic时，只有正在监听该topic地址的sub能够接收到消息；如果没有sub在监听，该topic就丢失了	Sender发送消息到目标Queue，receiver可以异步接收这个Queue上的消息。Queue上的消息如果暂时没有receiver来取，也不会丢失
消息发布接收策略	一对多的消息发布接收策略，监听同一个topic地址的多个sub都能收到publisher发送的消息。Sub接收完通知mq服务器	一对一的消息发布接收策略，一个sender发送的消息，只能有一个receiver接收。receiver接收完后，通知mq服务器已接收，mq服务器对queue里的消息采取删除或其他操作

ActiveMQ入门案例

- 使用ActiveMQ作为实现JMS中间件优点

- 1.多种语言和协议编写客户端。语言: Java, C, C++, C#, Python, PHP。
- 2.完全支持JMS1.1和J2EE 1.4规范 (持久化,XA消息,事务)
- 3.对Spring的支持,ActiveMQ可以很容易内嵌到使用Spring的系统里面去,而且也支持Spring2.0的特性
- 4.完全支持JMS1.1和J2EE 1.4规范 (持久化,XA消息,事务)
- 5.通过了常见J2EE服务器(如 Geronimo,JBoss 4, GlassFish,WebLogic)的测试,其中通过JCA 1.5 resource adaptors的配置,可以让ActiveMQ可以自动的部署到任何兼容J2EE 1.4 商业服务器上
- 6.支持多种传送协议
- 7.从设计上保证了高性能的集群,客户端-服务器,点对点
- 8.支持Ajax
- 9.支持与Axis的整合
- 10.可以很容易得调用内嵌JMS provider,进行测试

ActiveMQ入门案例

- ActiveMQ关键词

ActiveMQConnectionFactory :

实现了jms的ConnectionFactory , Connection的工厂类

Connection :

JMS连接 , 和Java连接池的Connection差不多 Producer和Consumer用来和Broker通讯的

Session :

会话

Destination :

目的地 , 数据要发送到哪里或者从哪里取

MessageProducer :

生产者

MessageConsumer :

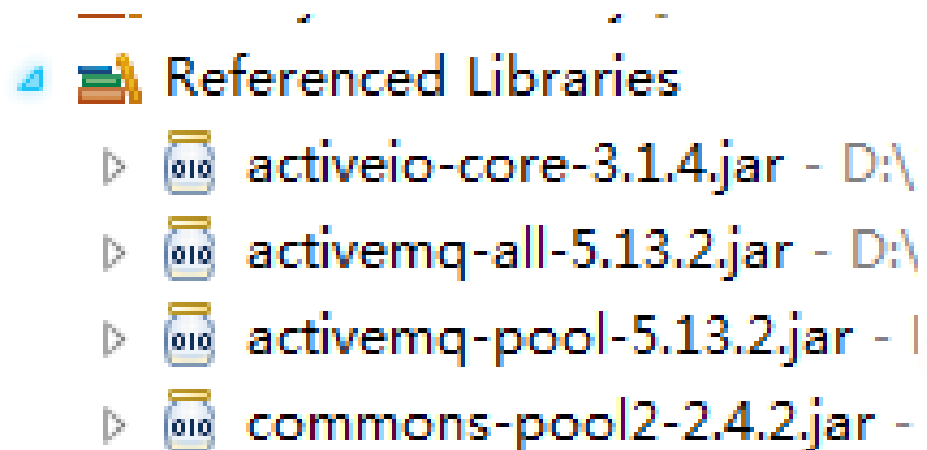
消费者

Message :

消息 , add到队列的东西 , 也就是自己要处理的东西 , Message有很多子接口 , TextMessage或ByteMessage

ActiveMQ入门案例

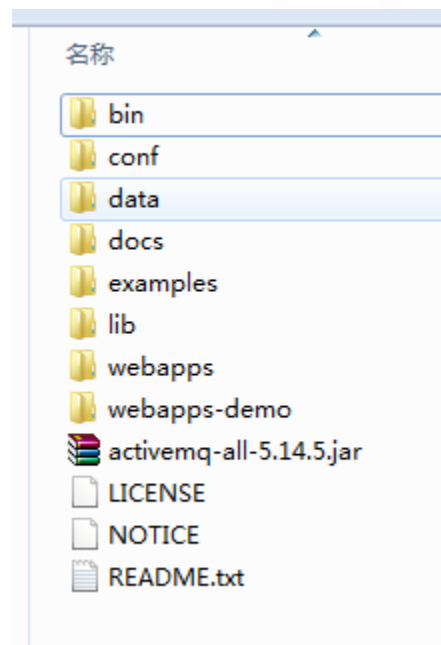
- 引入MQ相关jar包



ActiveMQ入门案例

• ActiveMQ目录说明

- bin:存放的是脚本文件
- conf:存放的是基本配置文件
- data:存放的是日志文件
- docs:存放的是说明文档
- examples:存放的是简单的实例
- lib:存放的是activemq所需jar包
- webapps:用于存放项目的目录



ActiveMQ入门案例-生产者

- Product.java

- // 默认连接用户名
- **private static final String USERNAME = ActiveMQConnection.DEFAULT_USER;**
- // 默认连接密码
- **private static final String PASSWORD = ActiveMQConnection.DEFAULT_PASSWORD;**
- // 默认连接地址
- **private static final String BROKEURL = ActiveMQConnection.DEFAULT_BROKER_URL;**
- // 发送的消息数量
- **private static final int SENDNUM = 10;**

ActiveMQ入门案例-生产者

发送消息

```
public static void sendMessage(Session session,  
    MessageProducer messageProducer) throws Exception {  
    for (int i = 0; i < Product.SENDNUM; i++) {  
        // 创建一条文本消息  
        TextMessage message = session.createTextMessage("ActiveMQ 发送消息" + i);  
        System.out.println("发送消息 : Activemq 发送消息" + i);  
        // 通过消息生产者发出消息  
        messageProducer.send(message);  
    }  
}
```

ActiveMQ入门案例-生产者

Main方法

```
ConnectionFactory connectionFactory;// 连接工厂
Connection connection = null;// 连接
Session session;// 会话 接受或者发送消息的线程
Destination destination;// 消息的目的地
MessageProducer messageProducer;// 消息生产者
connectionFactory = new ActiveMQConnectionFactory(Product.USERNAME,Product.PASSWORD,
    Product.BROKEURL); // 实例化连接工厂
connection = connectionFactory.createConnection();// 通过连接工厂获取连接
connection.start();// 启动连接
session = connection.createSession(true, Session.AUTO_ACKNOWLEDGE); // 创建session
destination = session.createQueue("HelloWorld"); // 创建一个名称为HelloWorld的消息队列
messageProducer = session.createProducer(destination);// 创建消息生产者
sendMessage(session, messageProducer);// 发送消息
session.commit();
```

ActiveMQ入门案例-消费者

消费者：customer.java

```
private static final String USERNAME = ActiveMQConnection.DEFAULT_USER; // 默认连接用户名  
private static final String PASSWORD = ActiveMQConnection.DEFAULT_PASSWORD; // 默认连接密码  
private static final String BROKEURL = ActiveMQConnection.DEFAULT_BROKER_URL; // 默认连接地址
```

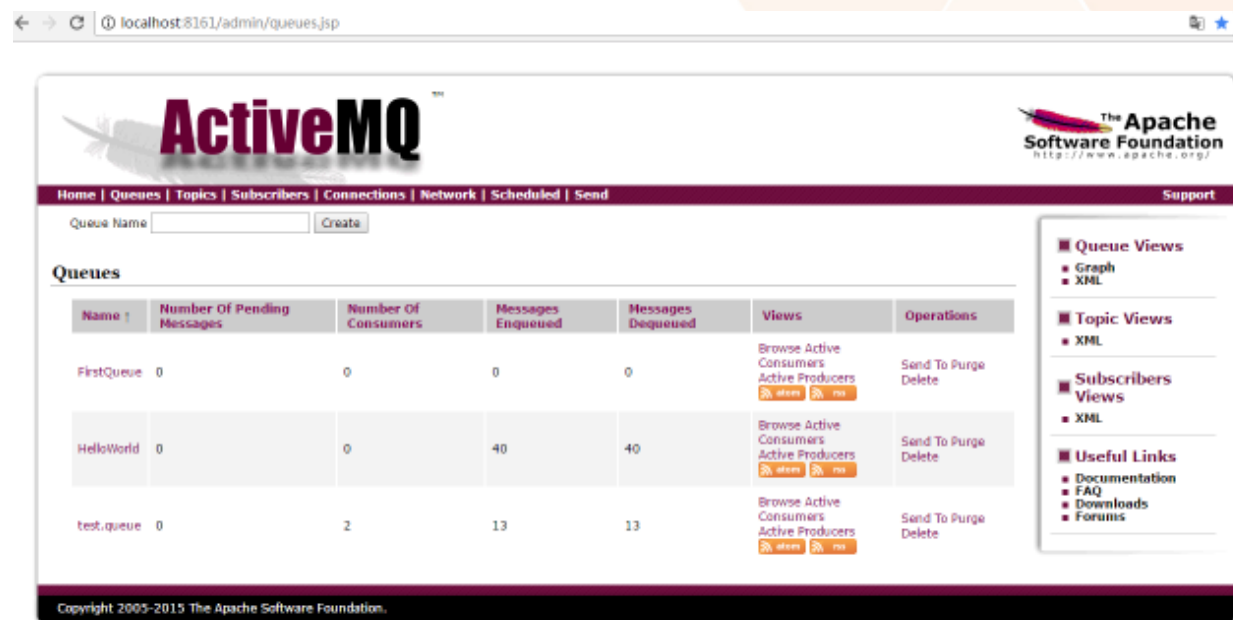

ActiveMQ入门案例-消费者

Main方法：

```
ConnectionFactory connectionFactory;// 连接工厂
Connection connection = null;// 连接
Session session;// 会话 接受或者发送消息的线程
Destination destination;// 消息的目的地
MessageConsumer messageConsumer;// 消息的消费者
// 实例化连接工厂
connectionFactory = new ActiveMQConnectionFactory(Consumer.USERNAME, Consumer.PASSWORD, Consumer.BROKEURL);
connection = connectionFactory.createConnection(); // 通过连接工厂获取连接
connection.start();// 启动连接
session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE); // 创建session
destination = session.createQueue("HelloWorld"); // 创建一个连接HelloWorld的消息队列
messageConsumer = session.createConsumer(destination);// 创建消息消费者
while (true) {
    TextMessage textMessage = (TextMessage) messageConsumer.receive(100000);
    if (textMessage != null) {
        System.out.println("收到的消息:" + textMessage.getText());
    } else {
        break;
    }
}
```

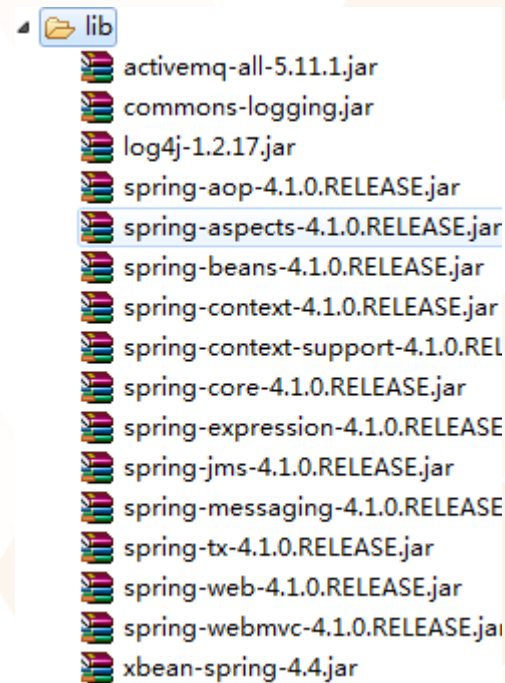
ActiveMQ入门案例-运行结果

- 启动MQ
- 监控MQ结果 （<http://localhost:8161/admin>）
- 启动cumstomer 启动product



ActiveMQ+SpringMVC集成

- 创建web工程
- 添加MQ和Spring的依赖包
- 配置web.xml、spring-mvc.xml
、 applicationContext.xml、 ActiveMQ.xml
- 创建生产者
- 创建消费者
- 监听



ActiveMQ+SpringMVC集成

ActiveMQ.xml

- 配置ActiveMQ的连接工厂
- 配置Spring 的Cache工厂
- 定义JmsTemplate的Queue类型
- 定义JmsTemplate的Topic类型
- 定义Queue监听
- 定义Topic监听

ActiveMQ+SpringMVC集成

ActiveMQ消息测试监控

- URL : <http://localhost:8161/admin/>
- 前端测试页面地址 : <http://localhost/ActiveMQSpringDemo/index.jsp>

常见协议

- 物理层：
 - 以太网·调制解调器·电力线通信(PLC)·SONET/SDH · 光导纤维等
- 数据链路层：
 - Wi-Fi(IEEE 802.11) · ATM · DTM · 令牌环·以太网· GPRS · HDLC · PPP · L2TP · PPTP · STP 等
- 网络层协议：
 - IP (IPv4 · IPv6) · ICMP· ICMPv6·IGMP ·IS-IS · ARP · RARP等
- 传输层协议：
 - TCP · UDP · TLS · DCCP · SCTP · RSVP 等
- 应用层协议：
 - HTTP · POP3 · RPC · DHCP · DNS · FTP · IMAP4 · IRC · XMPP · SMTP · SNMP · SSH · RTP · SOAP等

谢 谢