使用UDP广播事件

2018年7月25日

15:03

**UDP的基础知识**

UDP是无连接协议，没有持久化连接这样的概念，并且每个消息（一个UDP数据报）都是一个单独的传输单元。

UDP没有TCP的纠错机构，其中每个节点都将确认它们所接受到的包，而没有被确认的传输单包将会被发送方重新传输。

想打电话，消息双向流动。

**UDP广播**

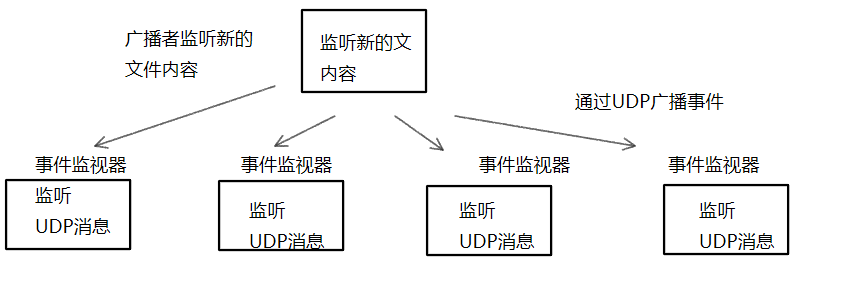
UDP除了单播外还提供例如向多个接收者发送额外消息的额外传输模式：

多播 —— 传输到一个预定义的主机组。

广播 —— 传输到网络（或者子网）上的所有主机

**UDP应用程序**

发布订阅模式



**消息POJO：LogEvent**

**package** xin.gonefuture.java.netty.UsingUDPBroadcastEvent;

**import** java.net.InetSocketAddress;

*/\*\**

*\** ***@author*** *gonefuture gonefuture@qq.com*

*\** ***@version*** *1.00*

*\* Description: JavaLearn*

*\* time 2018/7/25 15:56*

*\*/*

*/\**

*\* 日志POJO*

*\*/*

**public final class** LogEvent {

*// 分隔符*

**public static final byte *SEPARATOR*** = (**byte**) **':'**;

*// 源地址*

**private final** InetSocketAddress **source**;

*// 文件名*

**private final** String **logfile**;

*// 日志消息*

**private final** String **msg**;

*// 接受的时间戳*

**private final long received**;

*/\**

*用于传出消息的构造函数*

*\*/*

**public** LogEvent(String logfile, String msg) {

**this**(**null**, -1, logfile, msg);

}

**public** LogEvent(InetSocketAddress source, **long** received, String logfile, String msg) {

**this**.**source** = source;

**this**.**logfile** = logfile;

**this**.**msg** = msg;

**this**.**received** = received;

}

**public static byte** getSEPARATOR() {

**return *SEPARATOR***;

}

**public** InetSocketAddress getSource() {

**return source**;

}

**public** String getLogfile() {

**return logfile**;

}

**public** String getMsg() {

**return msg**;

}

*/\**

*返回接受LogEvent的时间*

*\*/*

**public long** getReceivedTimestamp() {

**return received**;

}

}

**编写广播者**

编写编码器

**package** xin.gonefuture.java.netty.UsingUDPBroadcastEvent;

**import** io.netty.buffer.ByteBuf;

**import** io.netty.channel.ChannelHandlerContext;

**import** io.netty.channel.socket.DatagramPacket;

**import** io.netty.handler.codec.MessageToMessageEncoder;

**import** io.netty.util.CharsetUtil;

**import** java.net.InetSocketAddress;

**import** java.util.List;

*/\*\**

*\** ***@author*** *gonefuture gonefuture@qq.com*

*\** ***@version*** *1.00*

*\* Description: JavaLearn*

*\** ***@time*** *2018/7/25 16:20*

*\*/*

**public class** LogEventEncoder **extends** MessageToMessageEncoder<LogEvent> {

**private final** InetSocketAddress **remoteAddress**;

*/\**

*LogEventEncoder创建了即将被发送到指定的InetSocketAddress的DatagramPacket消息*

*\*/*

**public** LogEventEncoder(InetSocketAddress remoteAddress) {

**this**.**remoteAddress** = remoteAddress;

}

@Override

**protected void** encode(ChannelHandlerContext ctx, LogEvent logEvent, List<Object> out) **throws** Exception {

**byte**[] file = logEvent.getLogfile().getBytes(CharsetUtil.***UTF\_8***);

**byte**[] msg = logEvent.getMsg().getBytes(CharsetUtil.***UTF\_8***);

ByteBuf buf = ctx.alloc().buffer(file.**length**+msg.**length**+1);

*// 将文件名写入*

buf.writeBytes(file);

*// 在文件名后面添加一个分隔符*

buf.writeByte(LogEvent.***SEPARATOR***);

*// 将日志消息写入*

buf.writeBytes(msg);

*// 将一个拥有数据和目的地地址的新Datagrampacket添加到出站的消息列表中*

out.add(**new** DatagramPacket(buf,**remoteAddress**));

}

}

编写启动类

**package** xin.gonefuture.java.netty.UsingUDPBroadcastEvent;

**import** io.netty.bootstrap.Bootstrap;

**import** io.netty.channel.Channel;

**import** io.netty.channel.ChannelOption;

**import** io.netty.channel.EventLoopGroup;

**import** io.netty.channel.nio.NioEventLoopGroup;

**import** io.netty.channel.socket.nio.NioDatagramChannel;

**import** java.io.File;

**import** java.io.RandomAccessFile;

**import** java.net.InetSocketAddress;

*/\*\**

*\** ***@author*** *gonefuture gonefuture@qq.com*

*\** ***@version*** *1.00*

*\* Description: JavaLearn*

*\** ***@time*** *2018/7/25 16:44*

*\*/*

**public class** LogEventBroadcaster {

**private final** EventLoopGroup **group**;

**private final** Bootstrap **bootstrap**;

**private final** File **file**;

**public** LogEventBroadcaster(InetSocketAddress address, File file) {

**this**.**group** = **new** NioEventLoopGroup();

**this**.**bootstrap** = **new** Bootstrap();

*// 引导该NioDatagramChannel（无连接的）*

**bootstrap**.group(**group**).channel(NioDatagramChannel.**class**)

*// 设置So\_BROADCAST套接字选项*

.option(ChannelOption.***SO\_BROADCAST***,**true**)

.handler(**new** LogEventEncoder(address));

**this**.**file** = file;

}

**public void** run() **throws** Exception {

Channel ch = **bootstrap**.bind(0).sync().channel();

**long** pointer = 0;

*// 启动主处理循环*

**for** (;;) {

**long** len = **file**.length();

**if** (len <pointer) {

*// file was reset 如果有必要，将文件指针设置到该文件的最后一个字节*

pointer =len;

} **else** {

*// Content was added*

RandomAccessFile raf = **new** RandomAccessFile(**file**,**"r"**);

*// 设置当前的文件指针，以确保没有任何的旧日志被发送*

raf.seek(pointer);

String line;

**while** ((line = raf.readLine()) != **null**) {

*// 对于每个日志条目，写入一个LogEvent到Channel中*

ch.writeAndFlush(**new** LogEvent(**null**, -1, **file**.getAbsolutePath(), line));

}

*// 储存其在文件中的当前位置*

pointer = raf.getFilePointer();

raf.close();

**try** {

*// 休眠1秒，如果被中断，则退出循环；否则重新处理它*

Thread.*sleep*(1000);

} **catch** (InterruptedException e) {

Thread.*interrupted*();

**break**;

}

}

}

}

**public void** stop() {

**group**.shutdownGracefully();

}

*/\**

*两个参数，端口号，文件路径*

*\*/*

**public static void** main(String... args) **throws** Exception {

**if** (args.**length** != 2) {

**throw new** IllegalArgumentException();

}

args[1] = **"C:\\Users\\gonefuture\\Desktop\\software\\shell\\test.log"**;

*// 启动后，在linux的shell里执行 `nc -l -u -p 9999` 监听UDP的广播*

LogEventBroadcaster broadcaster = **new** LogEventBroadcaster(

**new** InetSocketAddress(**"255.255.255.255"**, Integer.*parseInt*(args[0])), **new** File(args[1])

);

**try** {

broadcaster.run();

} **finally** {

broadcaster.stop();

}

}

}