client服务启动时，初始化GatewayTCPConnectionFactory，并且调用doStart()方法进行相关的初始化；

doStart()方法中一个步骤就是创建conn连接，步骤：

1、通过服务发现实例，查找注册的Netty server信息列表

2、针对每个Netty server，都创建gateway_client_num个conn连接；

GatewayTCPConnectionFactory

```
 * @author ohun@live.cn
 */
public class GatewayTCPConnectionFactory extends GatewayConnectionFactory {
    private final AttributeKey<String> attrKey = AttributeKey.valueOf("host_port");
    private final Map<String, List<Connection>> connections = Maps.newConcurrentMap();

    private ServiceDiscovery discovery;
    private GatewayClient gatewayClient;

    private MPushClient mPushClient;

    public GatewayTCPConnectionFactory(MPushClient mPushClient) {
        this.mPushClient = mPushClient;
    }

    @Override
    protected void doStart(Listener listener) throws Throwable {
        EventBus.register(this);

        gatewayClient = new GatewayClient(mPushClient);
        gatewayClient.start().join();
        discovery = ServiceDiscoveryFactory.create();
        discovery.subscribe(GATEWAY_SERVER, this);
        discovery.lookup(GATEWAY_SERVER).forEach(this::syncAddConnection);
        listener.onSuccess();
    }
```

同步创建连接

GatewayTCPConnectionFactory

```
    }

    private void syncAddConnection(ServiceNode node) {
        for (int i = 0; i < gateway_client_num; i++) {
            addConnection(node.getHost(), node.getPort(), true);
        }
    }

    private void addConnection(String host, int port, boolean sync) {
        ChannelFuture future = gatewayClient.connect(host, port);
        future.channel().attr(attrKey).set(getHostAndPort(host, port));
        future.addListener(f -> {
            if (!f.isSuccess()) {
                logger.error("create gateway connection failure, host={}, port={}", host, port, f.cause());
            }
        });
        if (sync) future.awaitUninterruptibly();
    }
```

1

利用一个bootstrap实例创建connect连接

NettyTCPClient connect()

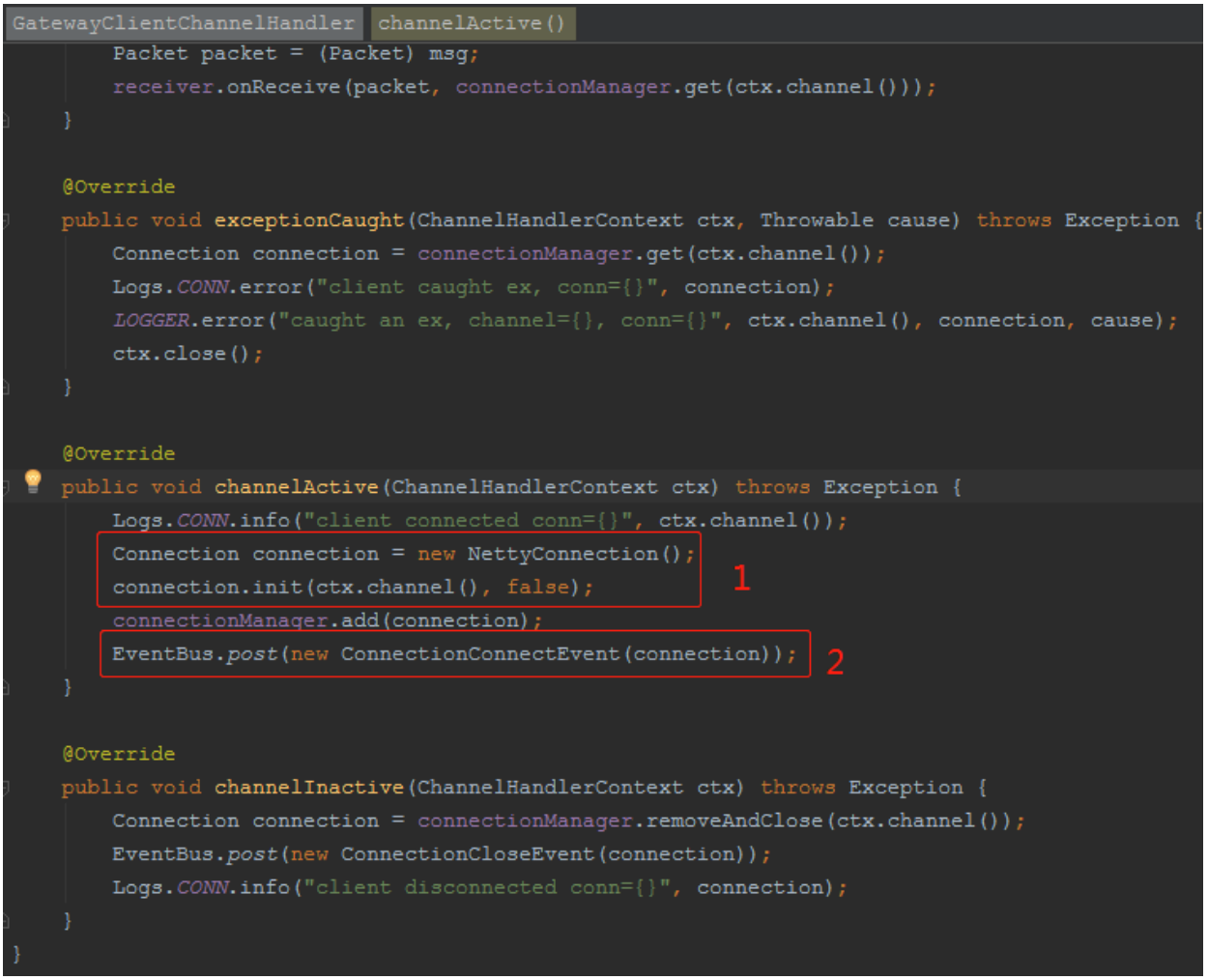
```
public abstract class NettyTCPClient extends BaseService implements Client {
    private static final Logger LOGGER = LoggerFactory.getLogger(NettyTCPClient.class);

    private EventLoopGroup workerGroup;
    protected Bootstrap bootstrap;

    private void createClient(Listener listener, EventLoopGroup workerGroup, ChannelFact
        this.workerGroup = workerGroup;
        this.bootstrap = new Bootstrap();
        bootstrap.group(workerGroup)//
                .option(ChannelOption.SO_REUSEADDR, true)//
                .option(ChannelOption.ALLOCATOR, PooledByteBufAllocator.DEFAULT)//
                .channelFactory(channelFactory);
        bootstrap.handler(new ChannelInitializer<Channel>() { // (4)
            @Override
            public void initChannel(Channel ch) throws Exception {
                initPipeline(ch.pipeline());
            }
        });
        initOptions(bootstrap);
        listener.onSuccess();
    }

    public ChannelFuture connect(String host, int port) {
        return bootstrap.connect(new InetSocketAddress(host, port));
    }
```

连接创建成功，会调用channelPipeLine中的GatewayClientChannelHandler#channelActive()方法




1、创建conn连接

2、通过事件总线EventBus，推送ConnectionConnectEvent事件给订阅了此事件的GatewayTCPConnectionFactory#on方法

GatewayTCPConnectionFactory

```
    private void syncAddConnection(ServiceNode node) {
        for (int i = 0; i < gateway_client_num; i++) {
            addConnection(node.getHost(), node.getPort(), true);
        }
    }

    private void addConnection(String host, int port, boolean sync) {
        ChannelFuture future = gatewayClient.connect(host, port);
        future.channel().attr(attrKey).set(getHostAndPort(host, port));
        future.addListener(f -> {
            if (!f.isSuccess()) {
                logger.error("create gateway connection failure, host={}, port={}", host, port, f.cause());
            }
        });
        if (sync) future.awaitUninterruptibly();
    }

    @Subscribe
    @AllowConcurrentEvents
    void on(ConnectionConnectEvent event) {
        Connection connection = event.connection;
        String hostAndPort = connection.getChannel().attr(attrKey).getAndSet(null); 1
        if (hostAndPort == null) {
            InetSocketAddress address = (InetSocketAddress) connection.getChannel().remoteAddress();
            hostAndPort = getHostAndPort(address.getAddress().getHostAddress(), address.getPort()); 2
        }
    3   connections.computeIfAbsent(hostAndPort, key -> new ArrayList<>(gateway_client_num)).add(connection);
        logger.info("one gateway client connect success, hostAndPort={}, conn={}", hostAndPort, connection);
    }

    private static String getHostAndPort(String host, int port) {
        return host + ":" + port;
    }
```

1、获取匹配属性channel中attrKey的值；

2、如果获取为空，则直接从conn对象中拿到IP和PORT；

3、将conn连接加入map里对应hostAndPort的List<Connection>中，也就是一个Netty Server对应一个conn池；