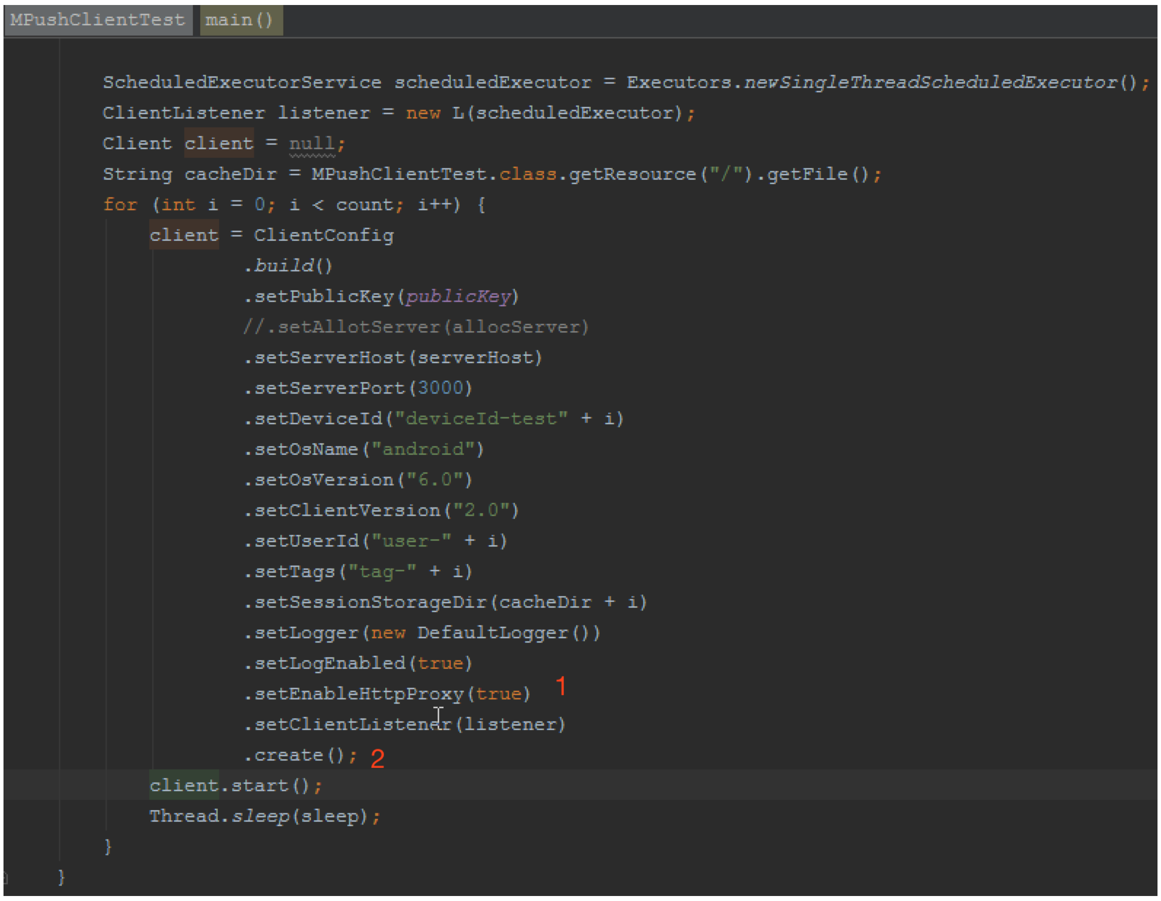
1、发送HTTP请求消息
2、接收HTTP响应消息

用SDK发送HTTP请求

客户端 MPushClient 提供了一个叫sendHttp的方法，该方法用于把客户端原本要通过HTTP方式发送的请求，全部通过PUSH通道转发，实现整个链路的长链接化；通过这种方式应用大大减少Http短链接频繁的创建，不仅仅节省电量，经过测试证明请求时间比原来至少缩短一倍，而且MPush提供的还有数据压缩功能，对于比较大的数据还能大大节省流量(压缩率4-10倍)，更重要的是所有通过代理的数据都是加密后传输的，大大提高了安全性！

1、设置ClientConfig.setEnableHttpProxy(true)来启用客户端代理。
2、通过Client.sendHttp(HttpRequest request)方法来发送请求。
AndroidSDK通过com.mpush.android.MPush#sendHttpProxy(HttpRequest request)来发送比较合适。

启动客户端代理
mpush-client-java工程，com/mpush/client/MPushClient.java
1、设置ClientConfig.setEnableHttpProxy(true)



```
ScheduledExecutorService scheduledExecutor = Executors.newSingleThreadScheduledExecutor();
ClientListener listener = new L(scheduledExecutor);
Client client = null;
String cacheDir = MPushClientTest.class.getResource("/").getFile();
for (int i = 0; i < count; i++) {
    client = ClientConfig
            .build()
            .setPublicKey(publicKey)
            //.setAllotServer(allocServer)
            .setServerHost(serverHost)
            .setServerPort(3000)
            .setDeviceId("deviceId-test" + i)
            .setOsName("android")
            .setOsVersion("6.0")
            .setClientVersion("2.0")
            .setUserId("user-" + i)
            .setTags("tag-" + i)
            .setSessionStorageDir(cacheDir + i)
            .setLogger(new DefaultLogger())
            .setLogEnabled(true)
            .setEnableHttpProxy(true)  1
            .setClientListener(listener)
            .create(); 2
    client.start();
    Thread.sleep(sleep);
}
}
```

ClientConfig#create()方法，初始化MPushClient实例

```
public Client create() {
 return new MPushClient(this);
}
```

MPushClient() 构造方法，具体实现细节，参考mpush-client-java下的《1 启动-建立连接.note》章节

```
MPushClient(ClientConfig config) {
 this.config = config;
 this.logger = config.getLogger();
 //初始化消息接收处理器（各种类型）
 MessageDispatcher receiver = new MessageDispatcher();
 //如果启用了代理，注册一个处理HTTP代理请求的处理器类 HttpProxyHandler
 if (config.isEnableHttpProxy()) {
 //HTTP 请求超时处理
 this.httpRequestMgr = HttpRequestMgr.I();
  receiver.register(Command.HTTP_PROXY, new HttpProxyHandler());
  }
  //ACK 超时处理
  this.ackRequestMgr = AckRequestMgr.I();
  //客户端conn连接管理
  this.connection = new TcpConnection(this, receiver);
this.ackRequestMgr.setConnection(this.connection);
}
```

MPushClient#start()方法

```
@Override
public void start() {
 if (clientState.compareAndSet(State.Shutdown, State.Started)) {
 connection.setAutoConnect(true);
 connection.connect();
 logger.w("do start client ...");
 }
}
```

发送请求

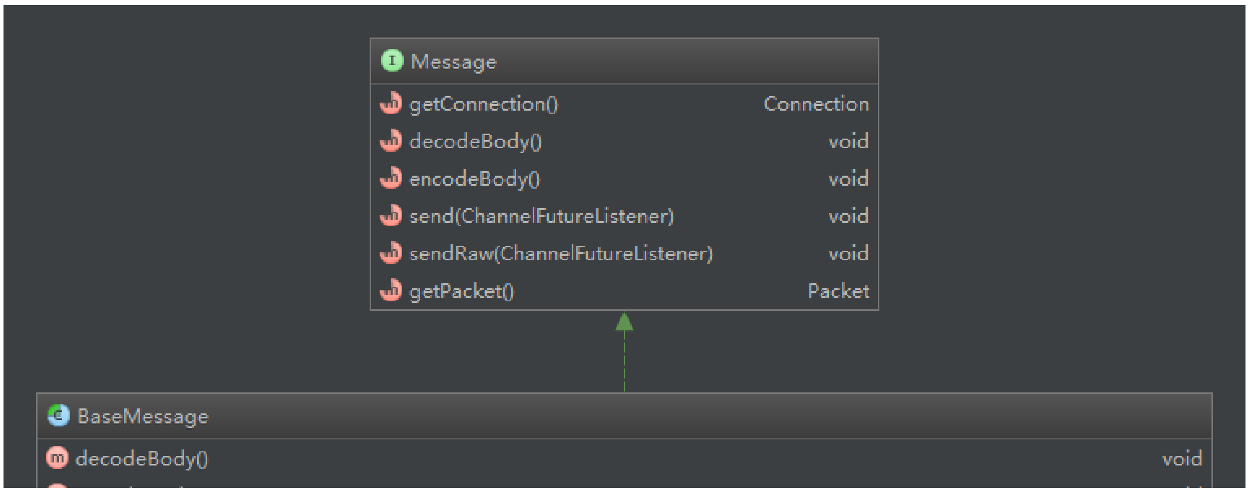
```
@Override
public Future<HttpResponse> sendHttp(HttpRequest request) {
 if (connection.getSessionContext().handshakeOk()) {
 HttpRequestMessage message = new HttpRequestMessage(connection);
 message.method = request.getMethod();
 message.uri = request.getUri();
 message.headers = request.getHeaders();
 message.body = request.getBody();
 message.send();
 logger.d("<<< send http proxy, request=%s", request);
 return httpRequestMgr.add(message.getSessionId(), request);
 }
 return null;
}
```

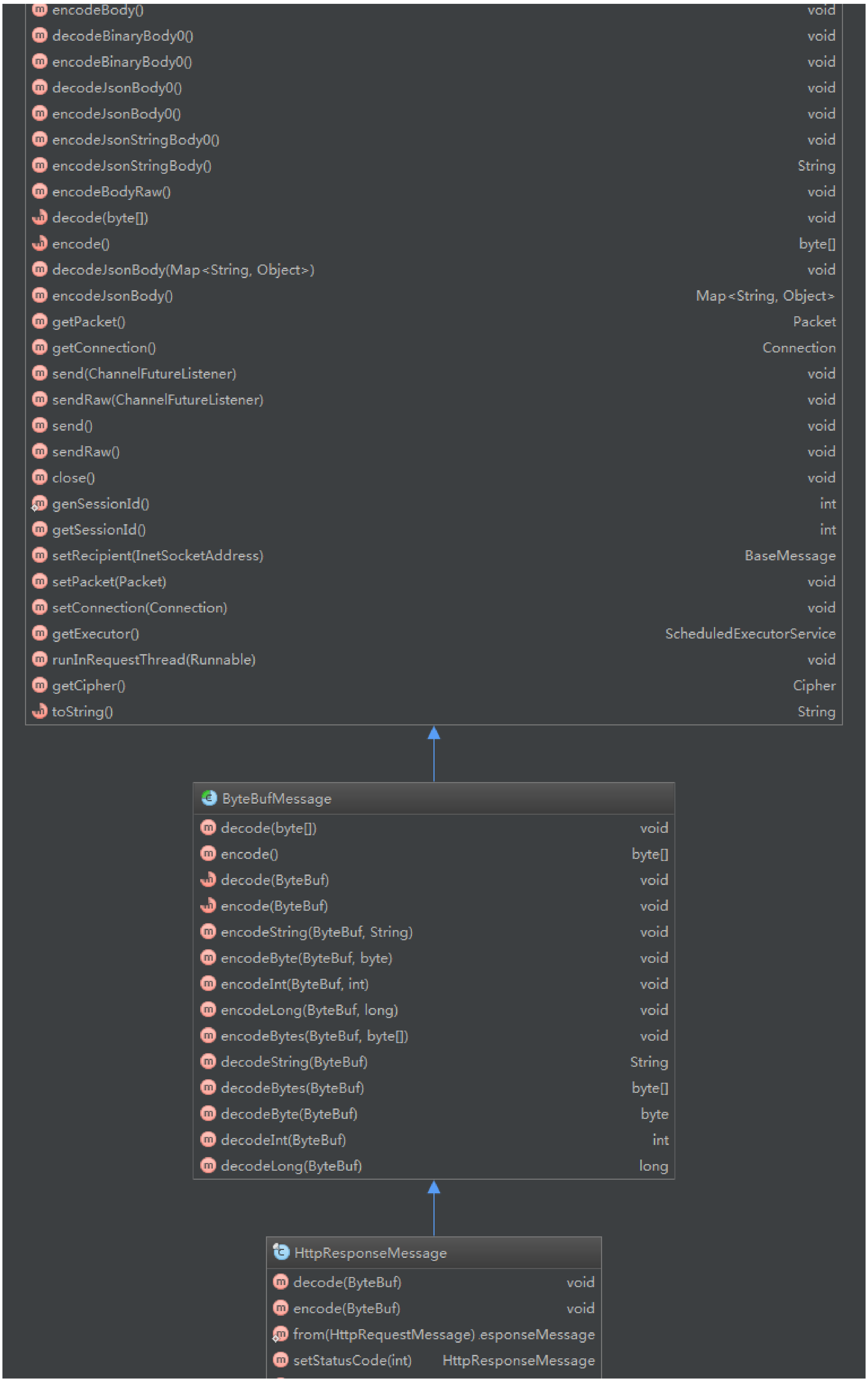
接收HTTP响应消息

```
//ConnClientChannelHandler.java

@Override
public void channelRead(ChannelHandlerContext ctx, Object msg) throws Exc
eption {
 connection.updateLastReadTime();
 if (msg instanceof Packet) {
 Packet packet = (Packet) msg;
 Command command = Command.toCMD(packet.cmd);
```

```java
if (command == Command.HANDSHAKE) {
 ...
 } else if (command == Command.FAST_CONNECT) {
 ...
 } else if (command == Command.KICK) {
 ...
 } else if (command == Command.ERROR) {
 ...
 } else if (command == Command.PUSH) {
 ...
 } else if (command == Command.HEARTBEAT) {
 ...
 } else if (command == Command.OK) {
 ...
 } else if (command == Command.HTTP_PROXY) {
 HttpResponseMessage message = new HttpResponseMessage(packet, connection);
 message.decodeBody();
 //TODO 根据状态码statusCode，做业务处理
 LOGGER.info("receive http response, message={}, body={}",
 message, message.body == null ? null : new String(message.body, Constants.UTF_8));
 }
 }
 LOGGER.debug("receive package={}, chanel={}", msg, ctx.channel());
}
```

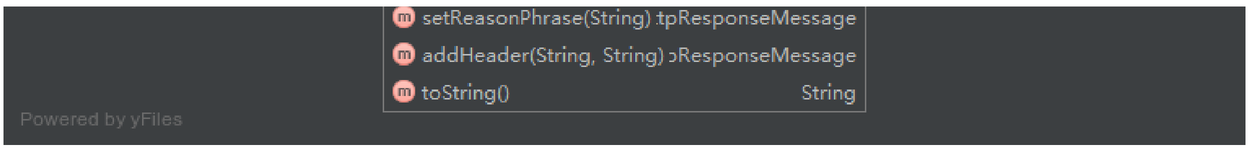

Message
getConnection() Connection
decodeBody() void
encodeBody() void
send(ChannelFutureListener) void
sendRaw(ChannelFutureListener) void
getPacket() Packet
BaseMessage
decodeBody() void



encodeBody()	void
decodeBinaryBody0()	void
encodeBinaryBody0()	void
decodeJsonBody0()	void
encodeJsonBody0()	void
encodeJsonStringBody0()	void
encodeJsonStringBody()	String
encodeBodyRaw()	void
decode(byte[])	void
encode()	byte[]
decodeJsonBody(Map<String, Object>)	void
encodeJsonBody()	Map<String, Object>
getPacket()	Packet
getConnection()	Connection
send(ChannelFutureListener)	void
sendRaw(ChannelFutureListener)	void
send()	void
sendRaw()	void
close()	void
genSessionId()	int
getSessionId()	int
setRecipient(InetSocketAddress)	BaseMessage
setPacket(Packet)	void
setConnection(Connection)	void
getExecutor()	ScheduledExecutorService
runInRequestThread(Runnable)	void
getCipher()	Cipher
toString()	String

ByteBufMessage	
decode(byte[])	void
encode()	byte[]
decode(ByteBuf)	void
encode(ByteBuf)	void
encodeString(ByteBuf, String)	void
encodeByte(ByteBuf, byte)	void
encodeInt(ByteBuf, int)	void
encodeLong(ByteBuf, long)	void
encodeBytes(ByteBuf, byte[])	void
decodeString(ByteBuf)	String
decodeBytes(ByteBuf)	byte[]
decodeByte(ByteBuf)	byte
decodeInt(ByteBuf)	int
decodeLong(ByteBuf)	long

HttpResponseMessage	
decode(ByteBuf)	void
encode(ByteBuf)	void
from(HttpRequestMessage)	.esponseMessage
setStatusCode(int)	HttpResponseMessage


setReasonPhrase(String) tpResponseMessage
addHeader(String, String) ɔResponseMessage
toString() String
Powered by yFiles

```
public final class HttpResponseMessage extends ByteBufMessage {
 public int statusCode;
 public String reasonPhrase;
 public Map<String, String> headers = new HashMap<>();
 public byte[] body;

 public HttpResponseMessage(Packet message, Connection connection) {
 super(message, connection);
 }
  @Override
  public void decode(ByteBuf body) {
  statusCode = decodeInt(body);
  reasonPhrase = decodeString(body);
  headers = Utils.headerFromString(decodeString(body));
  this.body = decodeBytes(body);
  }
  @Override
  public void encode(ByteBuf body) {
  encodeInt(body, statusCode);
  encodeString(body, reasonPhrase);
  encodeString(body, Utils.headerToString(headers));
  encodeBytes(body, this.body);
  }
  public static HttpResponseMessage from(HttpRequestMessage src) {
  return new HttpResponseMessage(src.packet.response(HTTP_PROXY), src.con
nection);
  }
  public HttpResponseMessage setStatusCode(int statusCode) {
  this.statusCode = statusCode;
  return this;
  }
  public HttpResponseMessage setReasonPhrase(String reasonPhrase) {
  this.reasonPhrase = reasonPhrase;
  return this;
  }
  public HttpResponseMessage addHeader(String name, String value) {
```

```java
this.headers.put(name, value);
return this;
}
@Override
public String toString() {
return "HttpResponseMessage{" +
"statusCode=" + statusCode +
", reasonPhrase='" + reasonPhrase + '\'' +
", headers=" + headers +
", body=" + (body == null ? "" : body.length) +
'}';
}
}
```