t-io

技术白皮书

您的痛点， t-io 已阅已历

t-io 的惊喜，您且慢慢享受

目录

t-io简介

t-io使命

t-io性能

功能介绍

案例展示

t-io简介

t-io是基于java的网络框架和中台，它的使命是：

让天下没有难开发的网络程序

t-io是华人社区最具知名度的网络开发框架，完全开源，拥有完全自主的知识产权

t-io是第一个通过华为专业测试的国产开源网络框架

t-io使命

让天下没有难开发的网络程序

01.节约人力成本

毕业几个月的java工程师可胜任多年资深java工程师写的网络程序

02.降低硬件要求

t-io开发的程序仅需2核4G
非t-io开发的程序要16核32G

03.提升项目保障

强大的资源管理能力,使业务出现内存溢出和死锁风险概率降低99%

04.缩短开发时间

借助官方的tio-study可提升3倍以上的速度

t-io性能

t-io处理能力

- 用t-io写的程序每秒处理1000+万条消息
- 1.9G内存支撑30万TCP长连接

非t-io处理能力

- 非t-io开发的im每秒处理还在百万级徘徊
- 还在讨论如何让5000TCP长连接不宕机

测试现场一：https://www.tiocloud.com/41

测试现场二：https://www.tiocloud.com/61

t-io性能

每秒处理1000万+条消息

https://www.tiocloud.com/41

t-io性能

1.9G内存支撑30万TCP长连接 https://www.tiocloud.com/61

```
root@Rain-ubuntu:/home/ubuntu/software/nlife-1.0.0-release# free -m
              total        used        free      shared  buff/cache   available
Mem:           7433         778        5549         210        1106        6170
Swap:          7632           0        7632
root@Rain-ubuntu:/home/ubuntu/software/nlife-1.0.0-release# ss -s
Total: 674 (kernel 900)
TCP:   19 (estab 12, closed 0, orphaned 0, synrecv 0, timewait 0/0), port

Transport Total     IP        IPv6
*         900       -         -
RAW       1         0         1
UDP       6         4         2
TCP       19        14        5
INET      26        18        8
FRAG      0         0         0
```

t-io启动时内存占用率以及链接情况

```
root@Rain-ubuntu:/home/ubuntu/software/nlife-1.0.0-release# ss -s
Total: 50675 (kernel 50725)
TCP:   50018 (estab 50011, closed 0, orphaned 0, synrecv 0, timewait
0

Transport Total     IP        IPv6
*         50725     -         -
RAW       1         0         1
UDP       7         5         2
TCP       50018     13        50005
INET      50026     18        50008
FRAG      0         0         0

root@Rain-ubuntu:/home/ubuntu/software/nlife-1.0.0-release# free -m
              total        used        free      shared  buff/cache   available
Mem:           7433        1043        5107         210        1282        5752
Swap:          7632           0        7632
```

5W链接时内存占用情况

```
root@Rain-ubuntu:/home/ubuntu/software/nlife-1.0.0-release# ss -s
Total: 100674 (kernel 100750)
TCP:   100018 (estab 100011, closed 0, orphaned 0, synrecv 0, timewait 0/0), por
ts 0

Transport Total     IP        IPv6
*         100750    -         -
RAW       1         0         1
UDP       6         4         2
TCP       100018    13        100005
INET      100025    17        100008
FRAG      0         0         0

root@Rain-ubuntu:/home/ubuntu/software/nlife-1.0.0-release# free -m
              total        used        free      shared  buff/cache   available
Mem:           7433        1278        4699         210        1455        5365
Swap:          7632           0        7632
```

10W链接时内存占用情况

```
Total: 200673 (kernel 200725)
TCP:   200017 (estab 200010, closed 0, orphaned 0, synrecv 0, timewait 0/0), por
ts 0

Transport Total     IP        IPv6
*         200725    -         -
RAW       1         0         1
UDP       6         4         2
TCP       200017    13        200004
INET      200024    17        200007
FRAG      0         0         0

root@Rain-ubuntu:/home/ubuntu/software/nlife-1.0.0-release# free -m
              total        used        free      shared  buff/cache   available
Mem:           7433        1576        4055         210        1801        4784
Swap:          7632           0        7632
```

20W链接时内存占用情况

```
untu:/home/ubuntu/software/nlife-1.0.0-release# ss -s
3 (kernel 300725)
7 (estab 300010, closed 0, orphaned 0, synrecv 0, timewait 0/0)

tal     IP        IPv6
0725    -         -
        0         1
        4         2
0017    13        300004
0024    17        300007
        0         0

root@Rain-ubuntu:/home/ubuntu/software/nlife-1.0.0-release# free -m
              total        used        free      shared  buff/cache   available
Mem:           7433        1903        3348         244        2182        4160
Swap:          7632           0        7632
```

30W链接时内存占用情况

```
top - 12:04:19 up  1:39,  2 users,  load average: 0.57, 0.28, 0.10
Tasks:   1 total,   0 running,   1 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.2 us,  0.1 sy,  0.0 ni, 99.7 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem :  7612144 total,  3110004 free,  2018228 used,  2483912 buff/cache
KiB Swap:  7816188 total,  7816188 free,        0 used.  4190884 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 6203 root      20   0 5054900 1.148g  17556 S   0.0 15.8   2:42.79 java
```

java进程统计情况

03

功能介绍

01

TCP服务器&客户端

TCP server & Client

t-io最被大家熟知的是实现了TCP服务器&客户端

使用文档：https://www.tiocloud.com/doc/tio/90

t-io作为TCP服务器，各个类的职责一目了然

编码/解码/处理

AioHandler.java

启动

TioServer.java/TioClient.java

各种事件处理

AioListener.java

TCP连接上下文

ChannelContext.java

实用方法

Tio.java

全局配置

TioConfig.java

HTTP服务器

t-io内部通过tio-http完整的实现和大量使用了HTTP 1.1

使用文档：https://www.tiocloud.com/doc/tio/126

POM引入

```
<dependency>
  <groupId>org.t-io</groupId>
  <artifactId>tio-http-server</artifactId>
  <version>3.7.0.v20201010-RELEASE</version>
</dependency>
```

Websocket服务器

Websocket server

t-io内部通过tio-websocket完整的实现和大量的使用了Websocket协议

使用文档：https://www.tiocloud.com/doc/tio/125

POM引入

```
<dependency>
  <groupId>org.t-io</groupId>
  <artifactId>tio-websocket-server</artifactId>
  <version>3.7.0.v20201010-RELEASE</version>
</dependency>
```

UDP服务器&客户端

UDP server & Client

t-io内置了UDP服务器&客户端

使用文档：https://www.tiocloud.com/doc/tio/127

- 把工程以maven的形式导入到eclipse后
- 运行org.tio.showcase.udp.server.ShowcaseUdpServerStarter启动udp服务器，启动成功后，日志如下

```
2018-12-30 21:08:23,764 INFO  org.tio.core.udp.UdpServer[136]: started tio udp server: 0.0.0.0:3000
```

- 运行org.tio.showcase.udp.client.UdpClientStarter
- 观察服务器端的console，会看到如下日志

```
2018-12-30 21:11:27,336 INFO  o.t.s.u.s.ShowcaseUdpHandler[29]: 收到来自127.0.0.1:62699的消息:【775008、用tio开发udp，有点意思】
2018-12-30 21:11:27,336 INFO  o.t.s.u.s.ShowcaseUdpHandler[29]: 收到来自127.0.0.1:62699的消息:【775009、用tio开发udp，有点意思】
2018-12-30 21:11:27,336 INFO  o.t.s.u.s.ShowcaseUdpHandler[29]: 收到来自127.0.0.1:62699的消息:【775010、用tio开发udp，有点意思】
2018-12-30 21:11:27,336 INFO  o.t.s.u.s.ShowcaseUdpHandler[29]: 收到来自127.0.0.1:62699的消息:【775011、用tio开发udp，有点意思】
2018-12-30 21:11:27,336 INFO  o.t.s.u.s.ShowcaseUdpHandler[29]: 收到来自127.0.0.1:62699的消息:【775012、用tio开发udp，有点意思】
2018-12-30 21:11:27,336 INFO  o.t.s.u.s.ShowcaseUdpHandler[29]: 收到来自127.0.0.1:62699的消息:【775013、用tio开发udp，有点意思】
2018-12-30 21:11:27,336 INFO  o.t.s.u.s.ShowcaseUdpHandler[29]: 收到来自127.0.0.1:62699的消息:【775014、用tio开发udp，有点意思】
```

- 整个工程才3个类，简单到极致，少年加油吧！


```
2018-12-30 21:11:27,336 INFO  o.t.s.u.s.ShowcaseUdpHandler[29]: 收到来自127.0.0.1:62699的消息:【775008、用tio开发udp，有点意思】
2018-12-30 21:11:27,336 INFO  o.t.s.u.s.ShowcaseUdpHandler[29]: 收到来自127.0.0.1:62699的消息:【775009、用tio开发udp，有点意思】
2018-12-30 21:11:27,336 INFO  o.t.s.u.s.ShowcaseUdpHandler[29]: 收到来自127.0.0.1:62699的消息:【775010、用tio开发udp，有点意思】
2018-12-30 21:11:27,336 INFO  o.t.s.u.s.ShowcaseUdpHandler[29]: 收到来自127.0.0.1:62699的消息:【775011、用tio开发udp，有点意思】
2018-12-30 21:11:27,336 INFO  o.t.s.u.s.ShowcaseUdpHandler[29]: 收到来自127.0.0.1:62699的消息:【775012、用tio开发udp，有点意思】
2018-12-30 21:11:27,336 INFO  o.t.s.u.s.ShowcaseUdpHandler[29]: 收到来自127.0.0.1:62699的消息:【775013、用tio开发udp，有点意思】
2018-12-30 21:11:27,336 INFO  o.t.s.u.s.ShowcaseUdpHandler[29]: 收到来自127.0.0.1:62699的消息:【775014、用tio开发udp，有点意思】
```

流量监控和统计

Traffic monitoring and statistics

t-io是目前唯一一个内置完整流量监控和统计的网络框架

ip	建链次数	上行业务包	处理业务包	下行业务包	上行字节	处理字节	下行字节	上行TCP包	解码异常次数
202.111.53.188 无锡市	10,826	11,085	11,085	37,208	11.41M	8.06M	19.54M	21,533	0
183.221.116.91 成都市	2,312	2,346	2,346	7,575	2.40M	1.68M	4.05M	4,612	0
58.62.167.1 广州市	2,209	2,282	2,282	10,299	1.88M	1.53M	4.75M	5,432	0
112.17.116.161 杭州市	2,033	2,887	2,887	43,619	1.84M	1.78M	13.05M	3,054	0
124.160.42.146 杭州市	1,820	2,173	2,173	9,014	2.78M	2.72M	3.26M	2,393	0
119.139.197.30 深圳市	1,411	1,518	1,518	9,112	1.39M	991.17K	3.75M	2,830	0
119.145.83.83 广州市	1,305	1,403	1,403	8,924	1.33M	949.80K	3.62M	2,659	0
60.186.191.140 杭州市	928	973	973	4,898	972.87K	671.69K	2.20M	1,887	0
59.61.172.231 福州市	907	1,079	1,079	3,471	611.95K	601.76K	1.48M	1,109	0
182.200.20.203 沈阳市	873	1,038	1,038	4,517	592.93K	569.22K	1.45M	1,096	0
	39219.00	46748.00	46748.00	397264.00	37217223.00	30116462.00	151527819.00	69466.00	0.00

流量监控和统计

单个TCP连接t-io实现了如下监控项

- 本次解码失败的次数
- 最近一次收到业务消息包的时间(一个完整的业务消息包，一部分消息不算)
- 最近一次发送业务消息包的时间(一个完整的业务消息包，一部分消息不算)
- 最近一次收到业务消息包的时间：收到字节就算
- 最近一次发送业务消息包的时间：发送字节就算
- ChannelContext对象创建的时间
- 第一次连接成功的时间
- 连接关闭的时间
- 进入重连队列时间
- 本连接已发送的字节数
- 本连接已发送的packet数
- 本连接已处理的字节数
- 本连接已处理的packet数
- 处理消息包耗时，单位：毫秒。拿这个值除以handledPackets，就是处理每个消息包的平均耗时
- 本连接已接收的字节数
- 本连接已接收了多少次TCP数据包
- 本连接已接收的packet数
- 心跳超时次数
- 平均每次TCP接收到的字节数，这个可以用来监控慢攻击，配置PacketsPerTcpReceive定位慢攻击
- 平均每次TCP接收到的业务包数，这个可以用来监控慢攻击，此值越小越有攻击嫌疑
- 处理packet平均耗时，单位：毫秒

流量监控和统计

全体TCP连接t-io实现了如下监控项

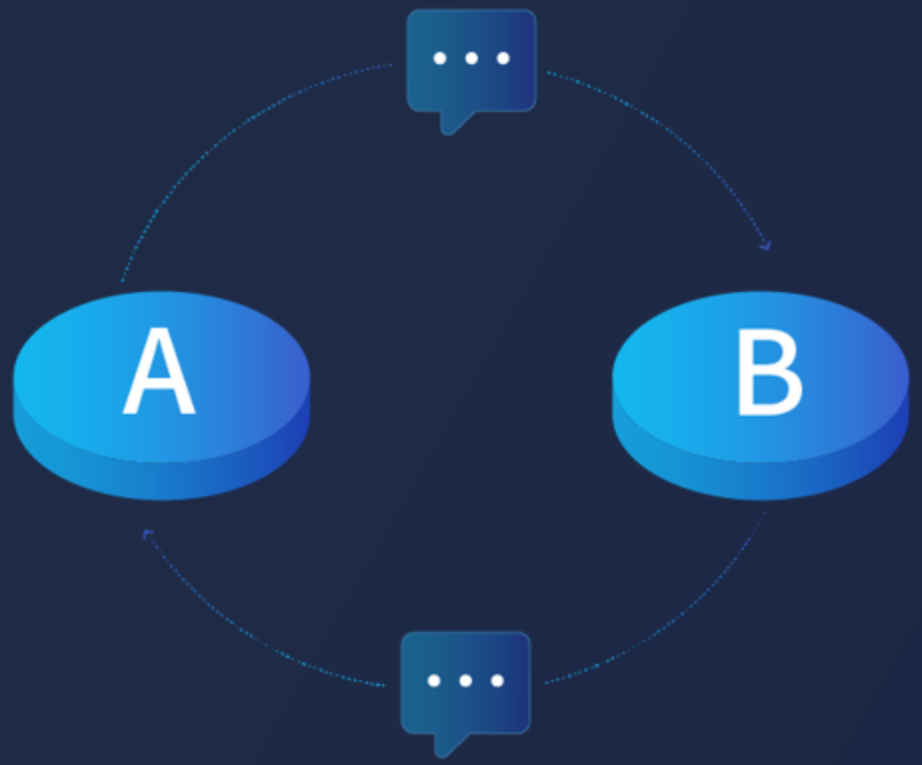
- 关闭了多少连接
- 接收到的消息包数
- 接收到的消息字节数
- 处理了的消息包数
- 处理消息包耗时，单位：毫秒
- 处理了多少字节
- 发送了的消息包数
- 发送了的字节数
- 本IP已接收了多少次TCP数据包
- 平均每次TCP接收到的字节数，这个可以用来监控慢攻击，配置PacketsPerTcpReceive定位慢攻击
- 平均每次TCP接收到的业务包数，这个可以用来监控慢攻击，此值越小越有攻击嫌疑
- 处理消息包耗时，单位：毫秒
- 处理packet平均耗时，单位：毫秒

同步消息
Synchronize messages

指A给B发送一条消息后，A在原线程中等待B给出一条对应的响应消息。

用其它网络框架实现这样的功能，需要开发人员精通多线程协作，极易产生死锁。
t-io已经内置了发送同步消息的能力，使用也非常简单。

参考文档：https://www.tiocloud.com/1301339231339290624



心跳检查
Heartbeat check

t-io内置了检测心跳超时的任务。

当发现某连接在指定时间内没有发生任何数据收发，
则断开该连接

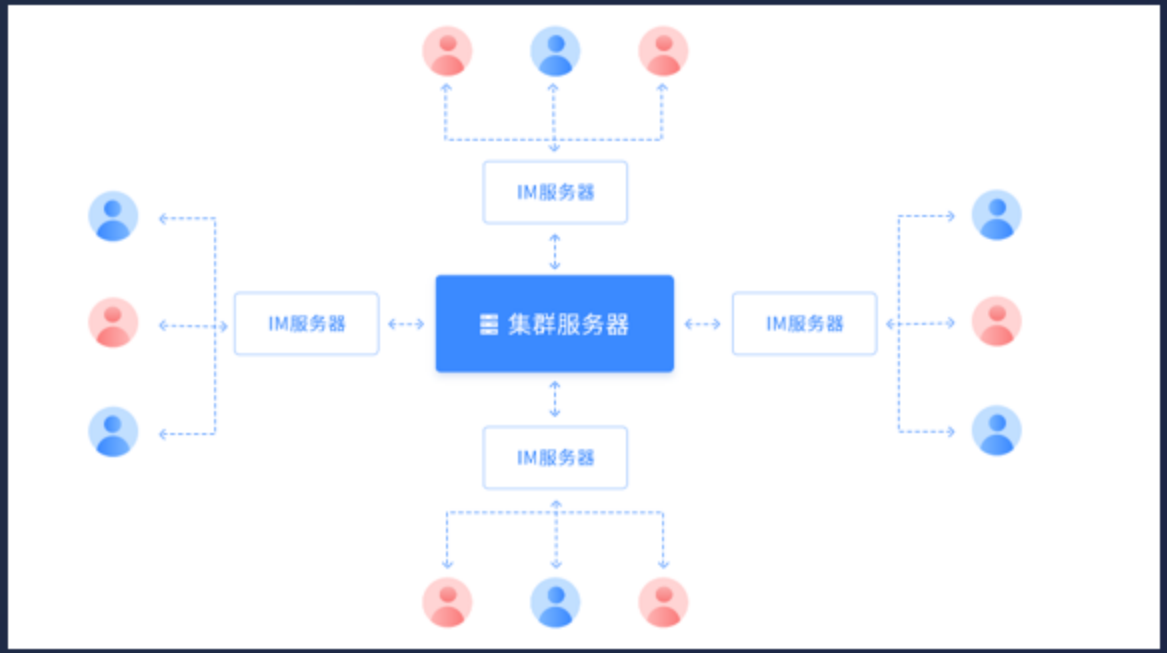
心跳发送
Heartbeat sending

t-io内置了发送心跳消息的任务

集群
Cluster

企业版t-io（集群版）内置了集群能力，用户只需要在启动的时候加几行配置代码，便能轻松拥有集群能力

在集群的赋能下，可同时支撑1亿+长连接



t-io集群：服务器路由


t-io集群服务器
业务服务器性能数据
RPC能力
Msg能力
ack能力
User路由表
Group路由表
token路由表
ip路由表
cid路由表
Bsid路由表
TCP监听
通知路由更新成功 7
6 通知更新路由数据
返回TCP业务服务器 3
2 获取TCP业务服务器
TCP连接池
绑定数据维护
Ack能力
RPC能力
Msg能力
t-io集群客户端
TCP业务服务器
TCP连接池
绑定数据维护
Ack能力
RPC能力
Msg能力
t-io集群客户端
HTTP业务服务器
Nginx
5 与选中的TCP业务服务器建立长连接
返回TCP业务服务器 4
1 获取TCP业务服务器
客户端（浏览器/安卓/IOS/PC/鸿蒙/麒麟...）

t-io集群：消息路由

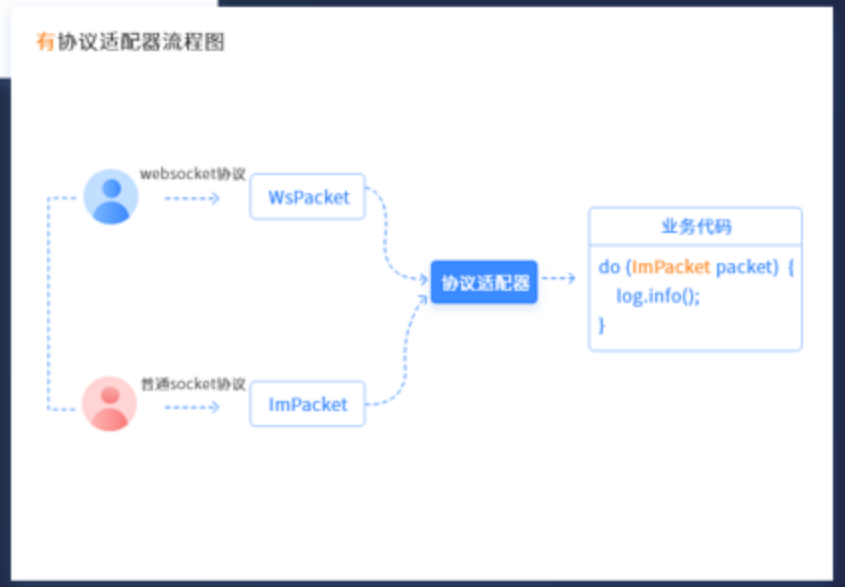

t-io集群服务器
业务服务器性能数据
RPC能力
Msg能力
ack能力
User路由表
Group路由表
token路由表
ip路由表
cid路由表
Bsid路由表
TCP监听
找到有group-1的tcp业务服务器，并通知其发消息给group-1
4
3
通知发消息给group-1
5
发消息给group-1
group-1的用户
TCP连接池
t-io集群客户端
TCP业务服务器
TCP连接池
t-io集群客户端
TCP业务服务器
2
发消息给group-1
group-1的用户
1
通知发消息给group-1
客户端（浏览器/鸿蒙/麒麟/安卓/IOS/PC...）

协议适配
Protocol adaptation

t-io内置了协议适配能力，让一套业务代码可同时支持多个协议。

无重复开发
无重复测试
无重复BUG

只需一套代码就可解决，
省开发、省测试、少BUG

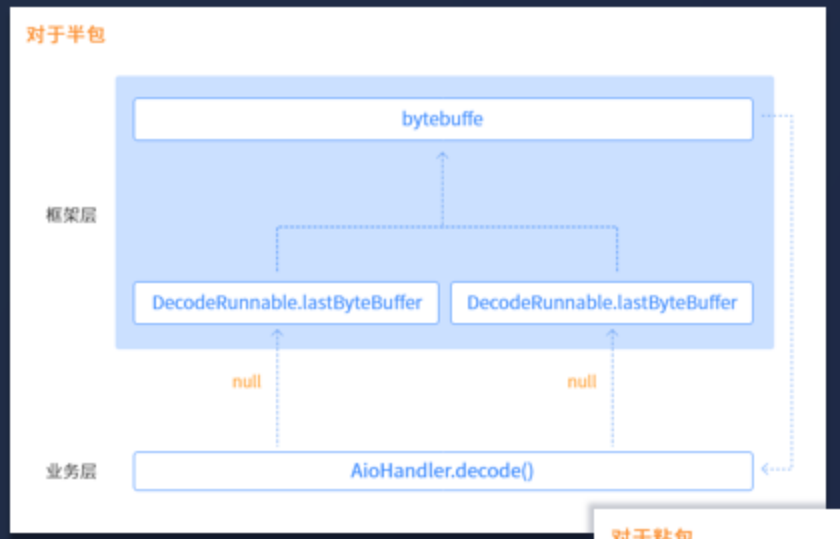


半包粘包

Half package

框架层已经做好半包和粘包的工作，业务层只需要按着业务协议解码即可

参考文档：https://www.tiocloud.com/doc/tio/132



如此循环，直到业务端能组成一个packet对象给框架层

自动重连
Automatic reconnection

用t-io作为TCP客户端时，内置了自动重连功能，用户只需要在启动时，配置ReconnConf对象即可。

- 如果业务自己实现自动重连，至少要浪费高级工程师5-10个工作日

SSL

t-io内置支持SSL，只需要一行代码：

```
serverTioConfig.useSsl("/xxx.jks", "/xxx.jks", "password");
```

IP拉黑

IP black

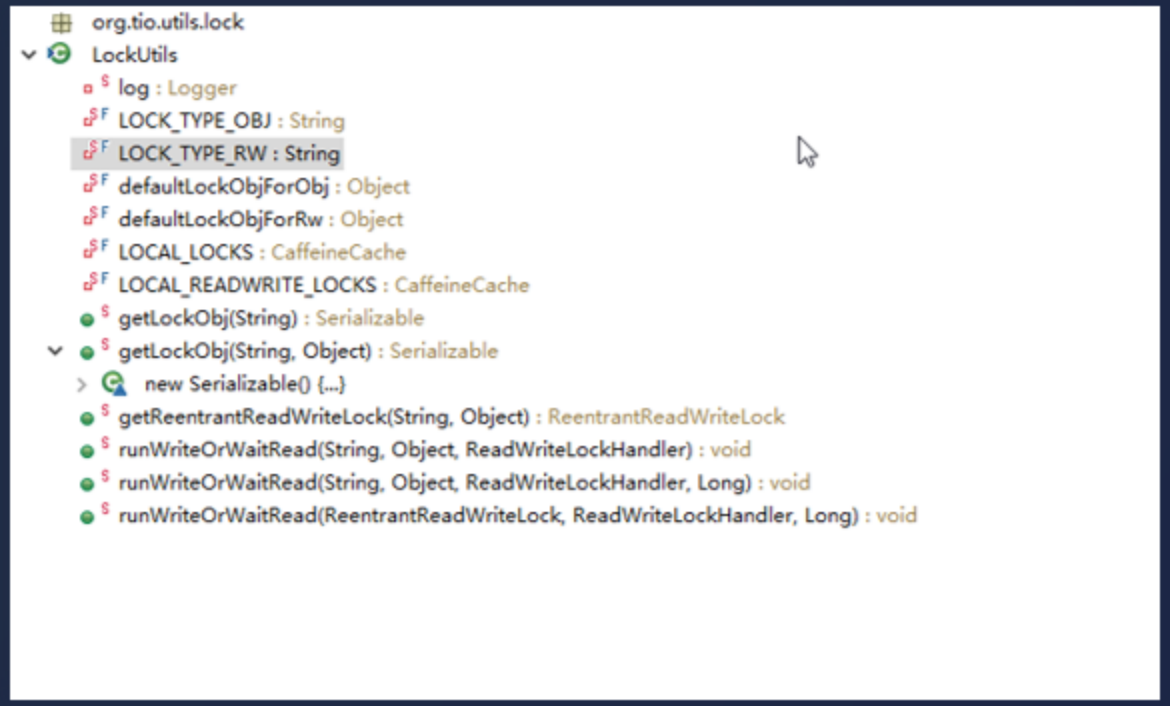
拉黑某个IP的时候，t-io会自动把该IP下的连接全部断开，并且不再接受来自该IP的连接，直到业务把该IP从黑名单中解除。

结合t-io的监控数据，可以很方便地实现自动拉黑能力

锁工具
Lock tool

为了更便捷地利用读写锁，t-io提供了一个被用户称为“出神出化”的锁工具类，将锁使用的出错率降至最低

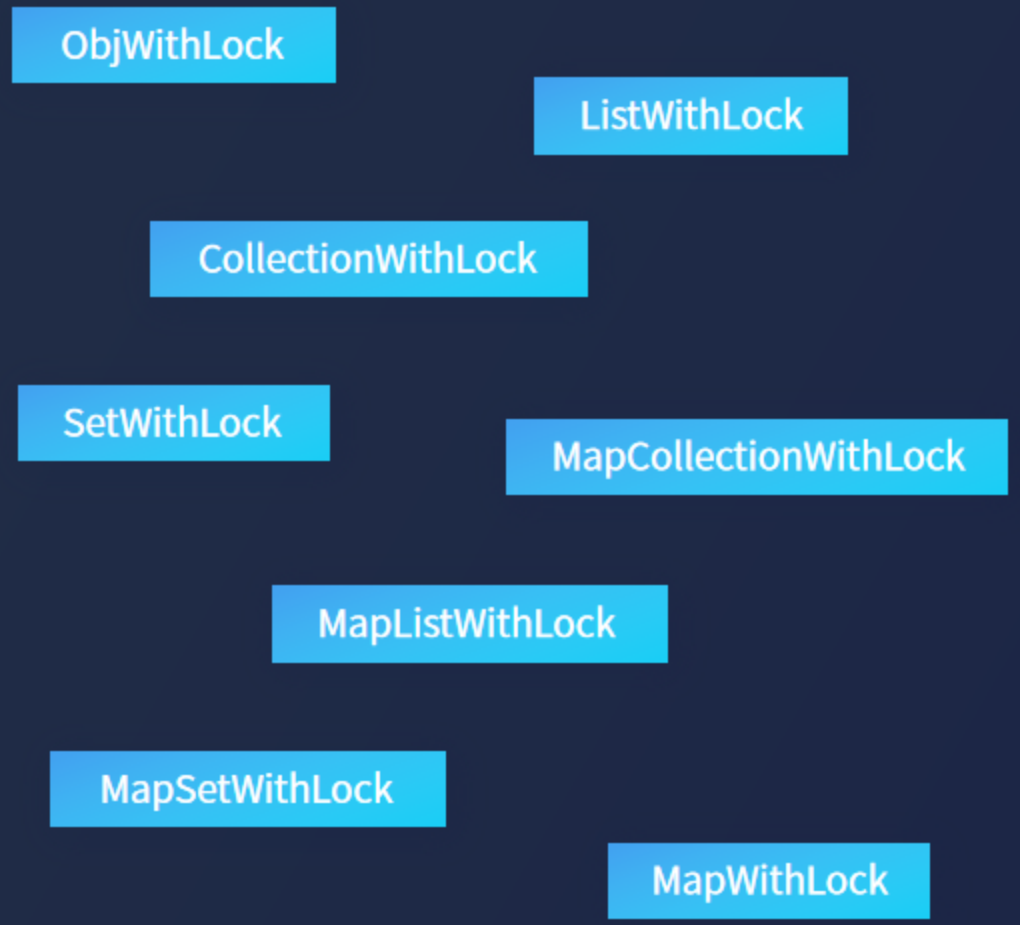
锁用不好，对项目就是灾难，我们的业务项目中大量使用这些t-io提供的锁工具，既高效又不易出错



并发数据结构

Protocol adapts concurrent data structure

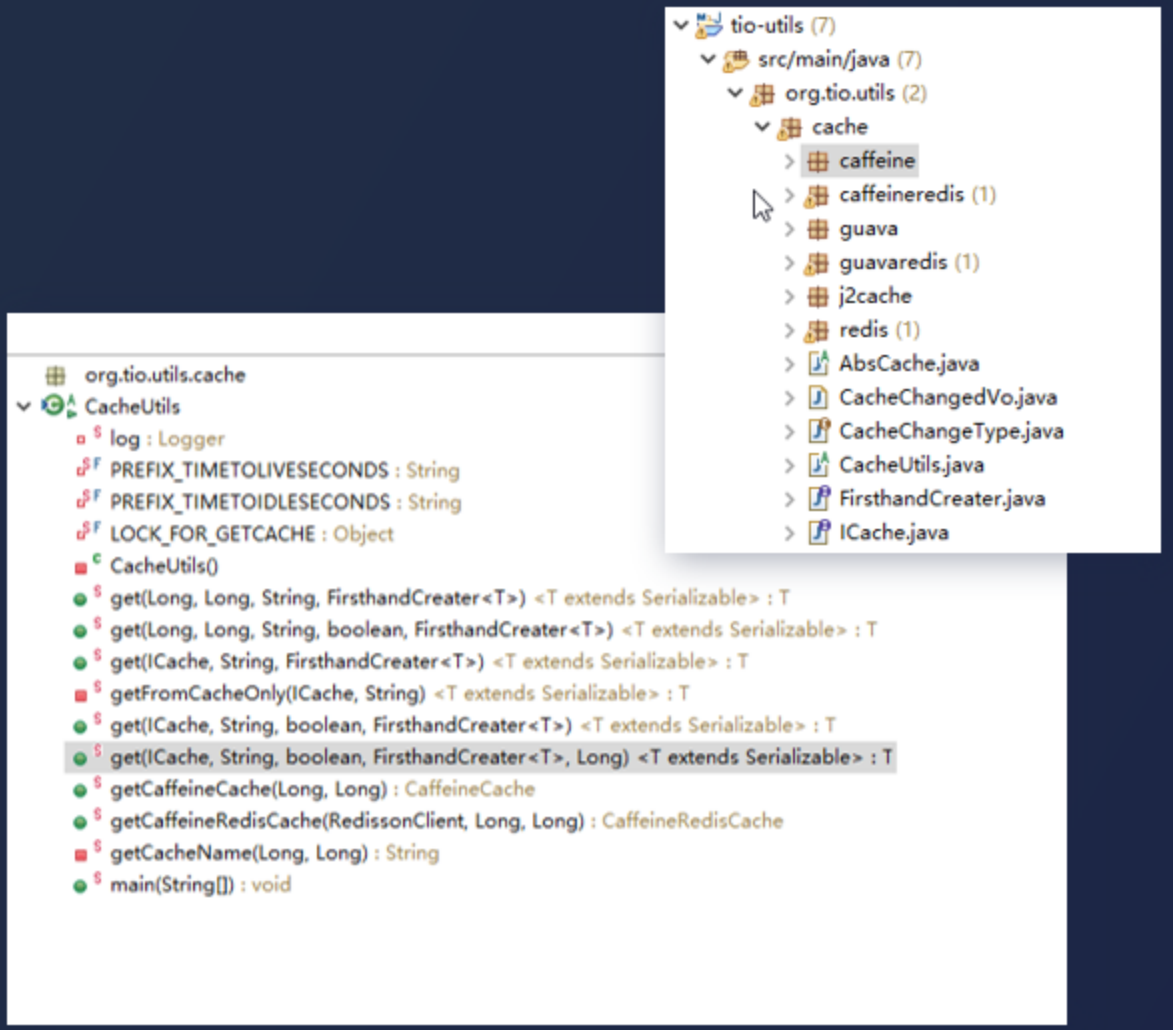
t-io自创了大量数据结构，将并发编程难度降到最低



缓存工具
Caching tools

t-io在第三方专业缓存的基础上，封装了两级缓存，同时提供了一个缓存工具类，用一套编程接口便可在多种缓存间自由切换。

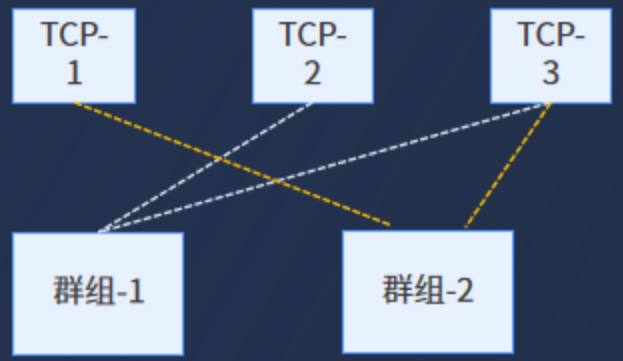
t-io官方产品大量使用了这些缓存工具，将性能发挥到极致



群组管理

group management

将一个或多个群组关联到一个或多个TCP（多对多）



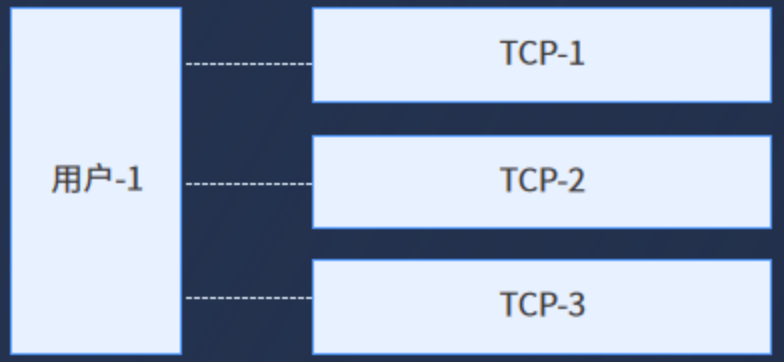
group

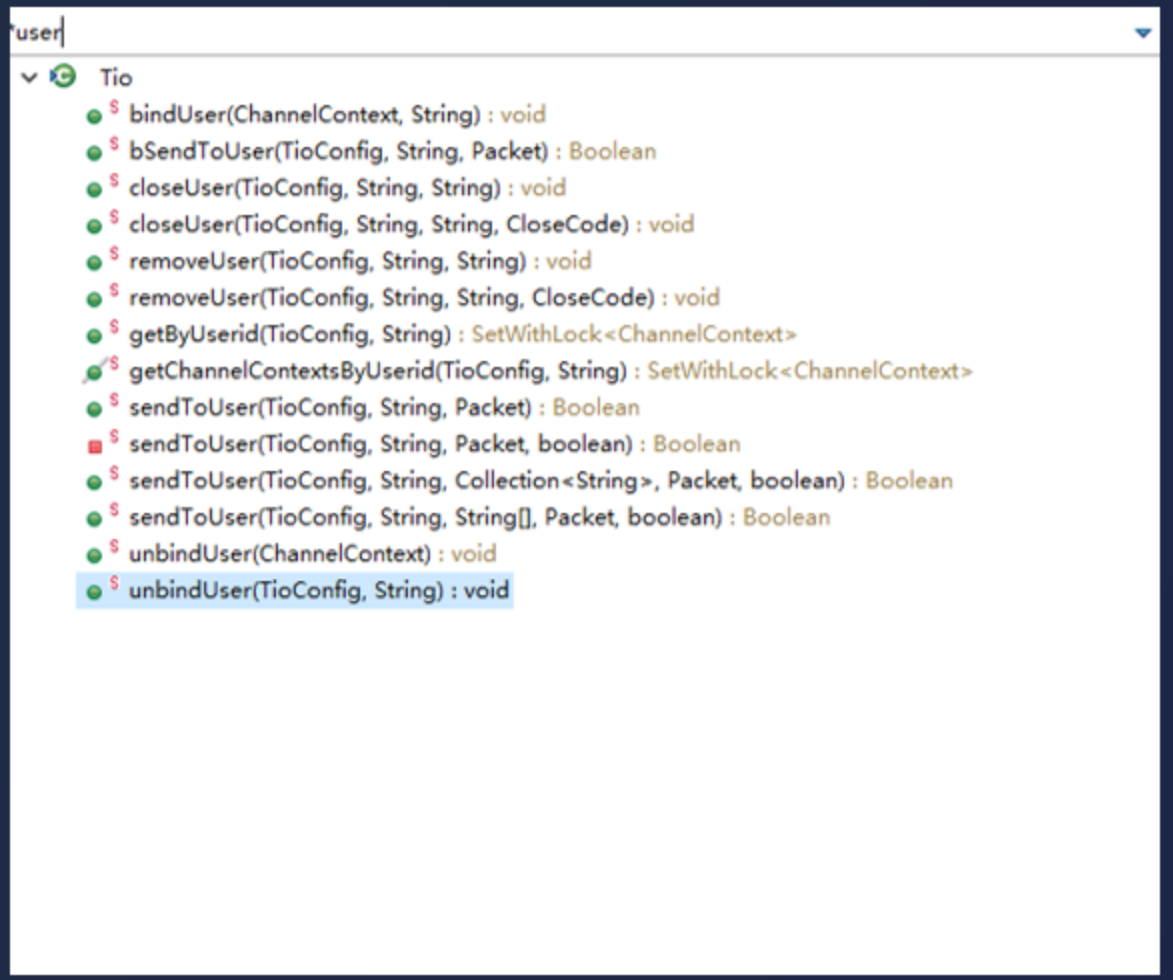
- Tio
  - bindGroup(ChannelContext, String) : void
  - bindGroup(TioConfig, String, String) : void
  - bSendToGroup(TioConfig, String, Packet) : Boolean
  - bSendToGroup(TioConfig, String, Packet, ChannelContextFilter) : Boolean
  - closeGroup(TioConfig, String, String) : void
  - closeGroup(TioConfig, String, String, CloseCode) : void
  - removeGroup(TioConfig, String, String) : void
  - removeGroup(TioConfig, String, String, CloseCode) : void
  - getByGroup(TioConfig, String) : SetWithLock<ChannelContext>
  - getChannelContextsByGroup(TioConfig, String) : SetWithLock<ChannelContext>
  - getPageOfGroup(TioConfig, String, Integer, Integer) : Page<ChannelContext>
  - getPageOfGroup(TioConfig, String, Integer, Integer, Converter<T>) <T> : Page<T>
  - groupCount(TioConfig, String) : int
  - isInGroup(String, ChannelContext) : boolean
  - sendToGroup(TioConfig, String, Packet) : void
  - sendToGroup(TioConfig, String, Packet, ChannelContextFilter) : void
  - sendToGroup(TioConfig, String, Packet, ChannelContextFilter, boolean) : Boolean
  - sendToGroup(TioConfig, String, Collection<String>, Packet, boolean) : Boolean
  - sendToGroup(TioConfig, String, String[], Packet, boolean) : Boolean
  - unbindGroup(ChannelContext) : void
  - unbindGroup(String, ChannelContext) : void
  - unbindGroup(TioConfig, String, String) : void

用户管理

user management

将一个用户关联到一个或多个TCP（一对多）

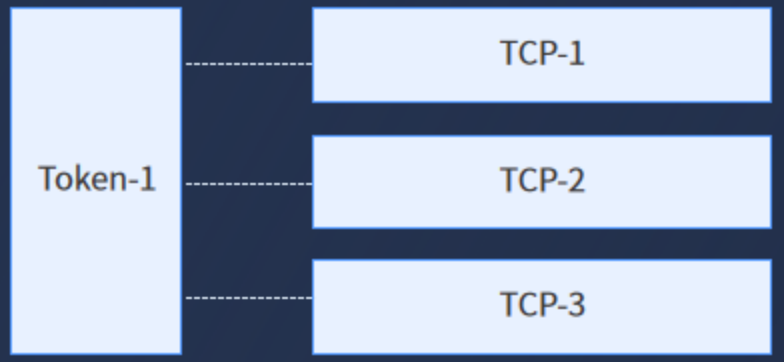


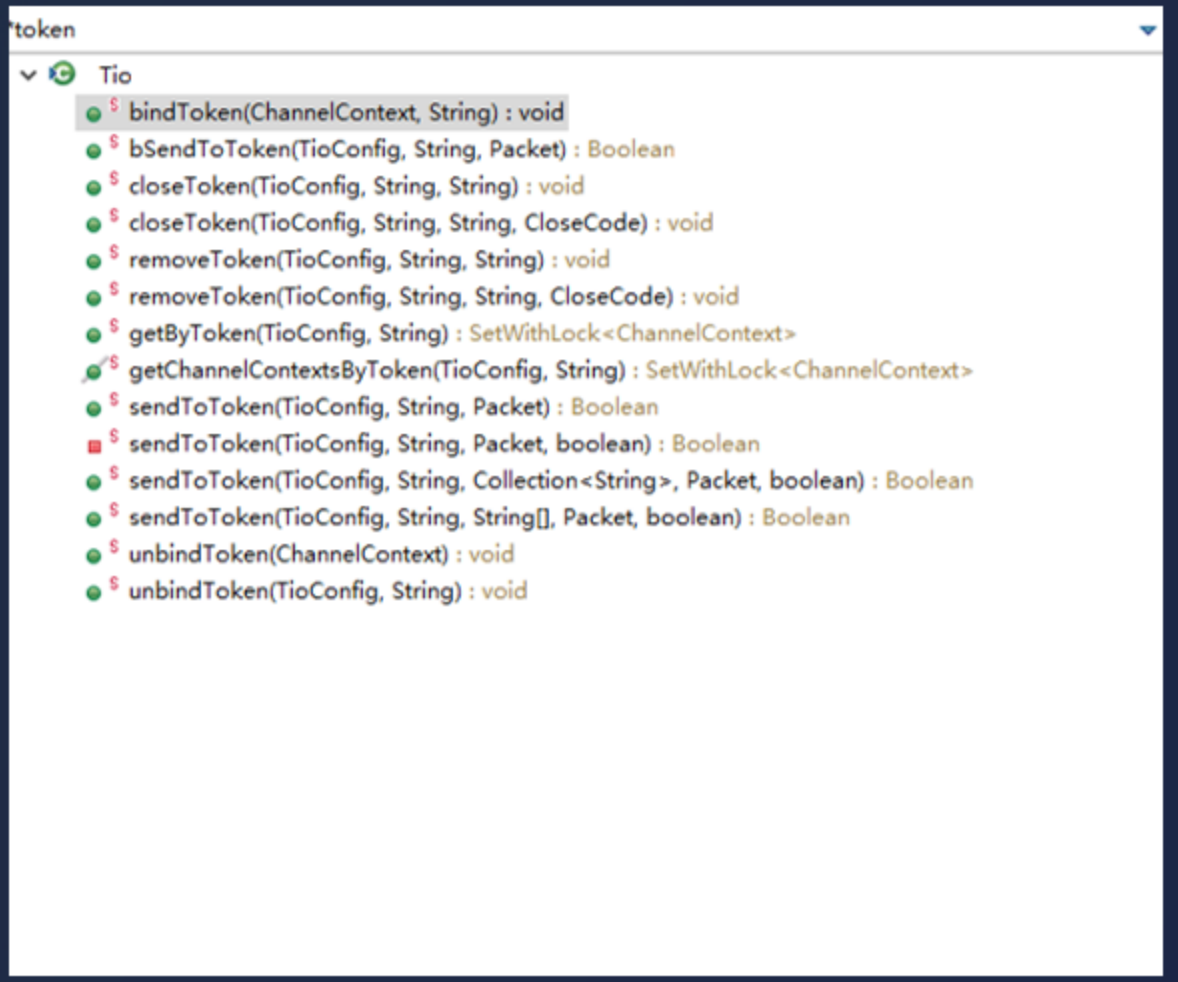


Token管理

Token management

将一个Token关联到一个或多个TCP（一对多）

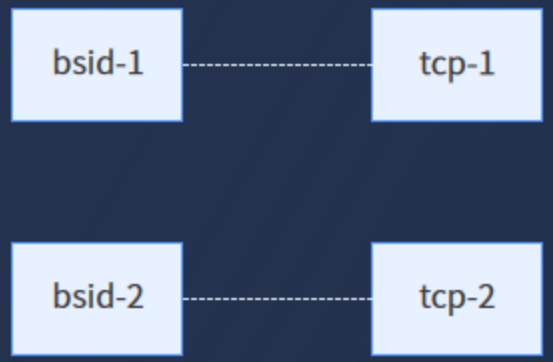


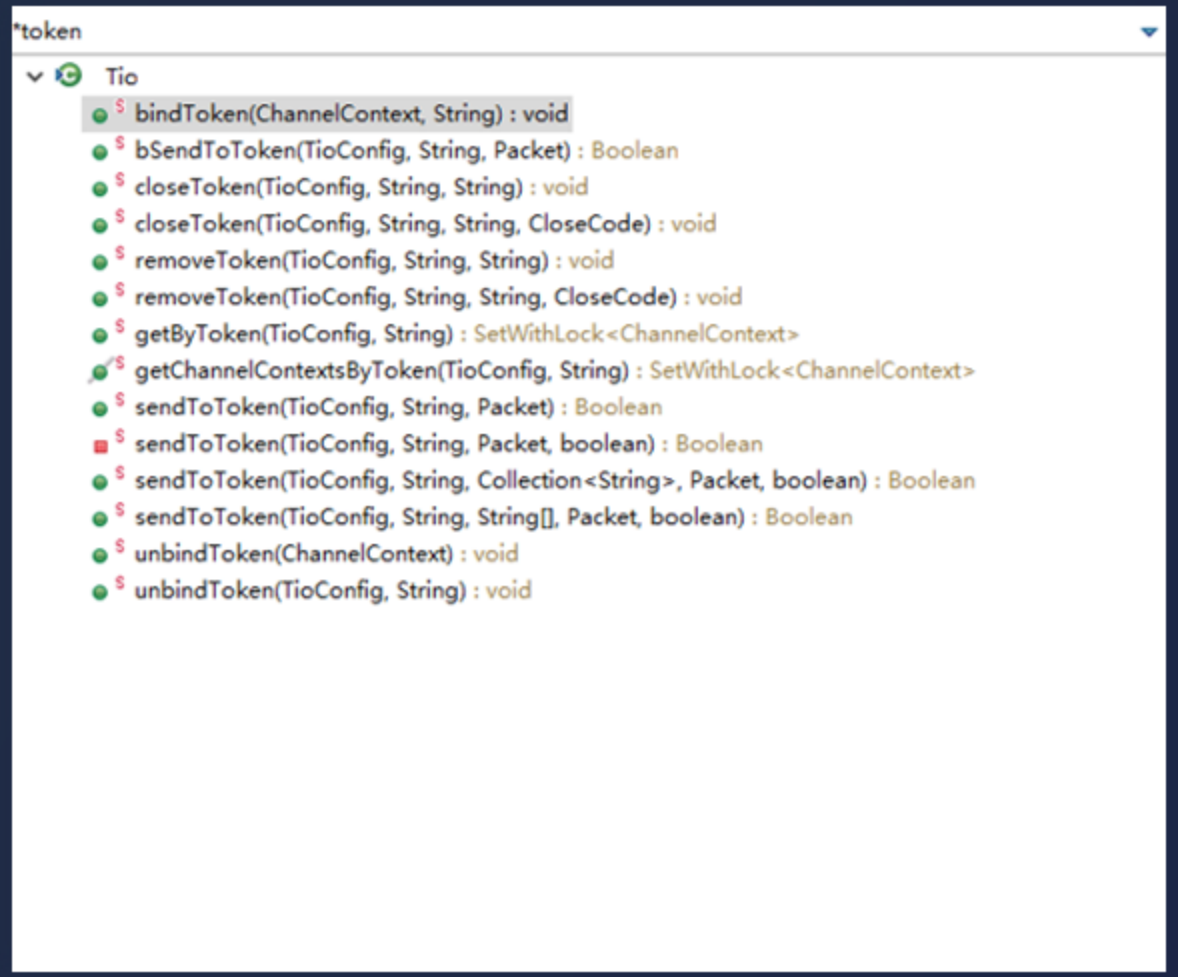


业务ID管理

Business ID management

将一个业务ID关联到一个TCP（一对一）

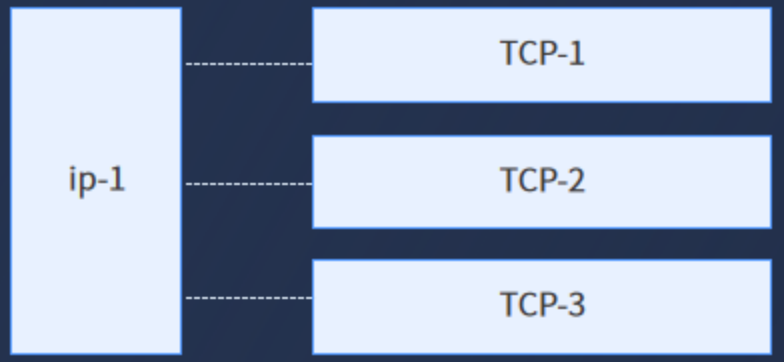


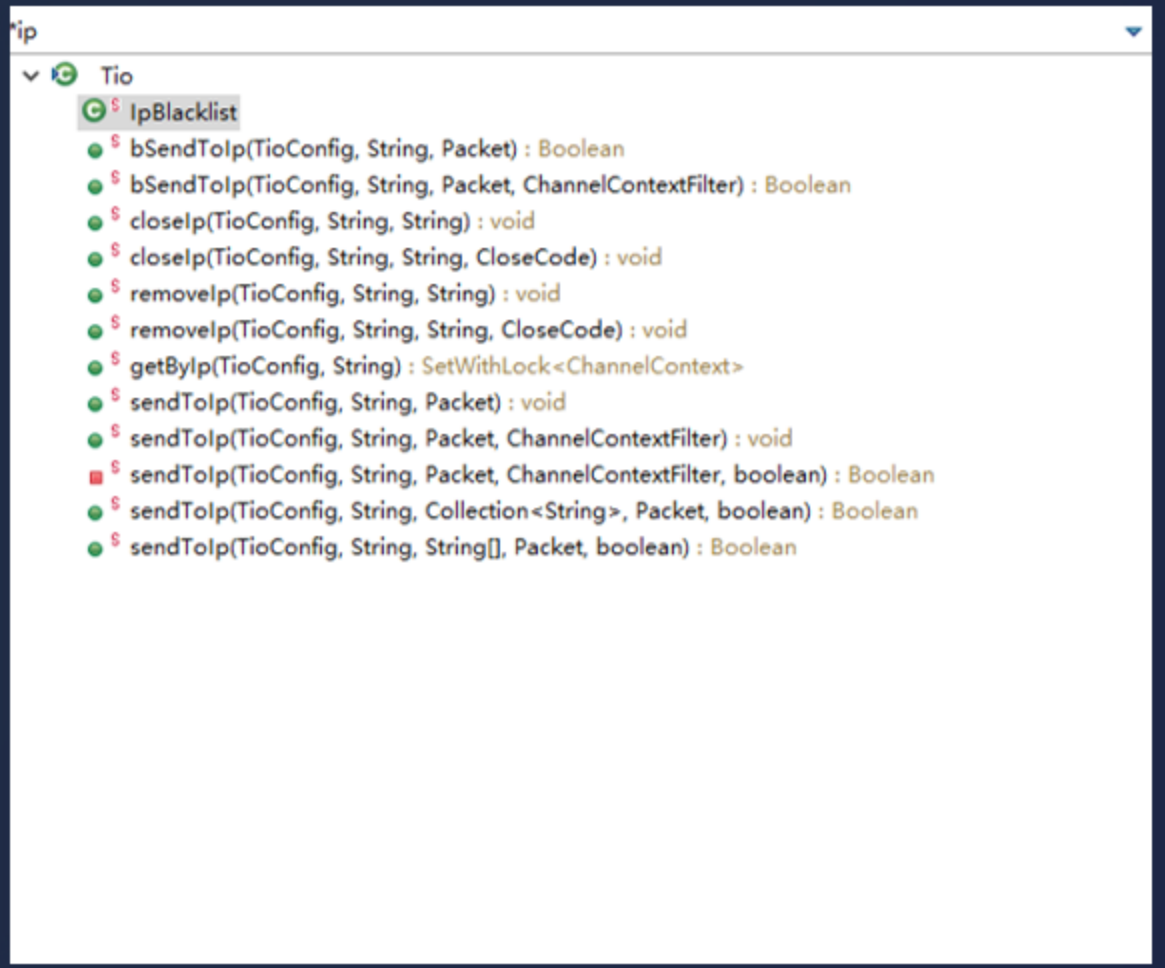


IP管理

IP management

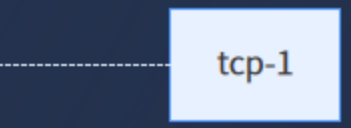
将一个客户端IP关联到一个或多个TCP（一对多）

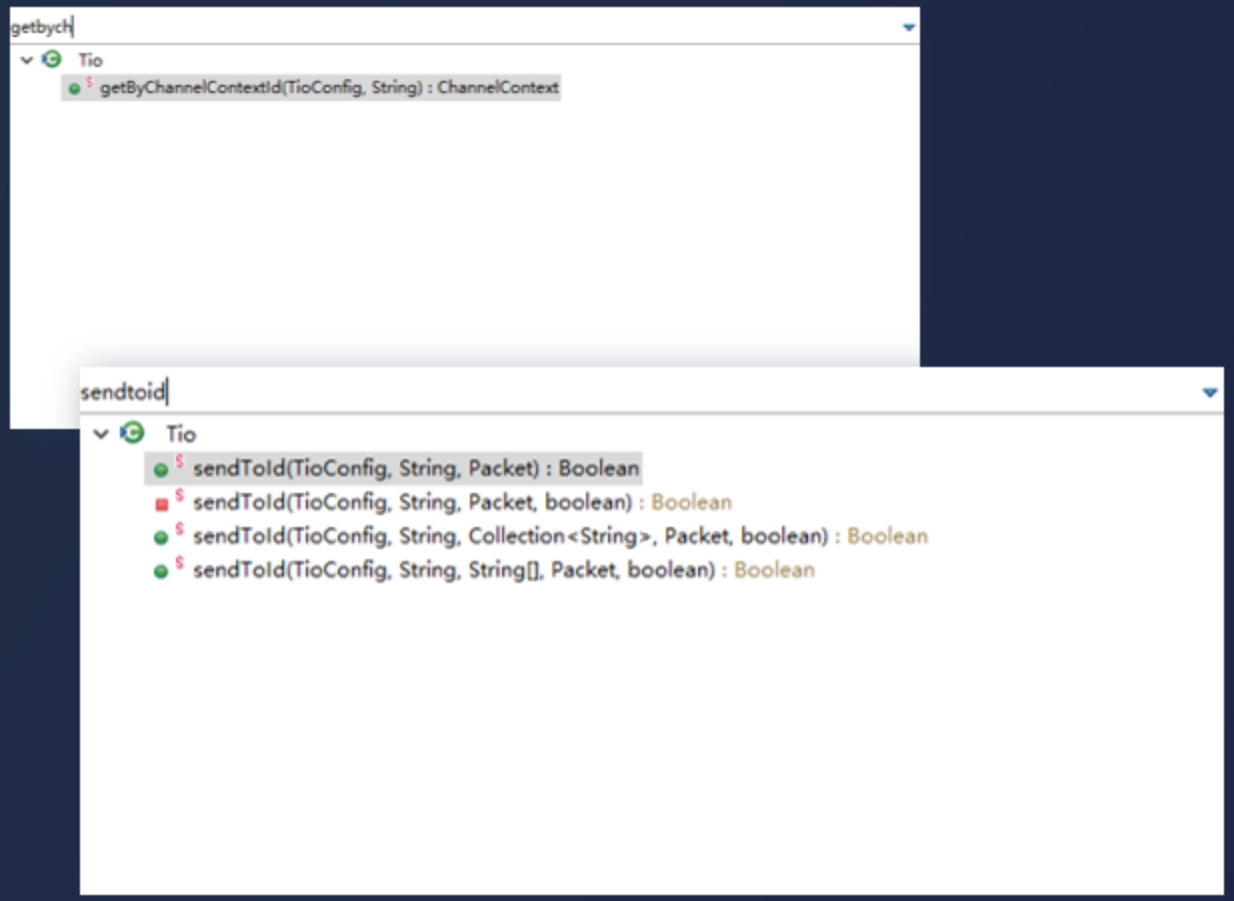




ID管理
ID management

将一个TCP ID关联到一个TCP（一对一）





05

案例介绍

t-io合作案例

保守估算，正在使用t-io的企业有4000+，并且还在不断增加中…

以下仅展示部分使用过t-io或购买过t-io授权的公司

更多案例：https://www.tiocloud.com/2/case/index.html

用户口碑（一）

> 如果没有t-io，华为这个智慧项目应该早就废了

睿泰数字 — 力挽狂澜

> 换上t-io后，吃嘛嘛香了（注：贝密游戏以前用的是netty）

贝密游戏 — 老板心安

> t-io短小精悍，但能解决大问题，有了t-io，网络编程不需要太多的精力投入，可以把更多的技术资源聚焦在产生商业价值的业务开发上

氮氪科技CTO胡鸿鹄 — 倚天屠龙

> 万幸有t-io，很多复杂的业务逻辑可以很简单的做出来，也希望t-io可以越来越好，造福更多的开发者和企业！

北京弈世教育科技发展有限公司 — 化繁为简

> t-io：秒上手，易监控，稳得狠！

如梦技术开源组织 — 易用稳定

> 没有t-io, 只能一辈子玩HTTP、JSON

EOVA开源社区 — 跨界利器

用户口碑（二）

用t-io搭建了公司所有服务的推送中心，能力杠杠滴....

河南迪信通商贸有限公司

无所不能

接入t-io后，快速实现了我们Groups需要的即时消息发送，效果棒棒滴!

上海航翼网络科技有限公司

高效稳定

t-io开源之初就关注了，非常有幸能够认识这么优秀的作者，无意间掌握了t-io的核心开发技术，现在让我稳坐技术头把交椅，再也回不去了

开源中国前工程师胡承淞

成就个人

t-io是我们从众多网络框架中精心挑选出来的，操作简单，稳定高效，做物联网相关产品简直如虎添翼!

蟋蟀物联网

碾压突围

t-io是一款高效、稳定、性能优异的网络框架。每一个基于t-io的项目就是妥妥的成功案例

越洋科技

成功的保障

用了t-io，项目的进度至少加快了半年，为我们的即时类数据提供了稳定、高效的解决方案

成都鼎融科技

极致效率

用户口碑（三）

早先用t-io做了个有即时能力的网站，也因此入职了开源中国。用了t-io后，我有更多的时间陪家人了

开源中国前工程师柯真

成就个人

t-io作为网络编程终极武器，处理各种通讯规约、私有协议得心应手！t-io值得每个网络开发者拥有！

赵传刚

事实标准

利用t-io,我们公司三年工作经验的开发人员就能搭建支撑上万台gps设备的物联网平台，非常感谢t-io的强大能力

江苏野马软件科技有限公司

强大无比

接触t-io之前，觉得即时通信这块很难，要投入很大的成本，接触t-io后，发现这块变得简单了，t-io化繁为简的能力太强大了

linkwechat创始人
江冬勤

化繁为简

t-io：网络编程的瑞士军刀，上手容易适配强！
t-io让我们团队可以集中优势力量攻克业务难关！

石家庄老顽童网络科技有限公司

万能军刀

朝闻t-io，夕出谭聊

南京皓叶腾信息科技有限公司

心想事成

t-io 历经十年打磨

网民笔伐过

用户捧杀过

华为极限压测过

最终化茧成蝶

为祖国信创添砖加瓦

谢谢欣赏！