

Network module

dtransmit

Socket（注意S大写） 现在机器人上这个类叫Foo

封装过的UDP socket，它的成员除了socket本身之外还包含回调函数readHandler、接收缓冲区recvBuffer、数据来源remoteEndpoint（ip和端口号）。

通过std::map，端口号可以对应到一个Socket，进而也可以对应到Socket的成员。

Socket(boost::asio::io_service &service, PORT port)

Socket的构造函数，传入io_service和port。socket的异步接收由service统一管理，socket绑定到本机端口port上。

Dtransmit

接收数据包：

void startRecv(PORT port, ReadHandler handler)

PORT就是int，ReadHandler是模板占位符，不用管

传入一个本机端口号和对应的readHandler，用这个端口号对应的socket开始接收数据包，把数据来源ip和端口号保存在Socket.remoteEndpoint中，并把数据交给readHandler处理。

这是异步操作，不阻塞主线程。

void addRawRecv(PORT port, std::function<void(void *, std::size_t)> callback)

传入一个本机端口号和回调函数callback，用端口号新建一个Socket，定义其readHandler功能为：只要从socket正常获取数据包，就把它交给回调函数callback。

随后调用startRecv，开始接收并处理数据包。

void addRawRecvFiltered(PORT port, std::string remoteEndpoint, std::function<void(void *, std::size_t)> callback)

传入一个本机端口号，数据包来源remoteEndpoint和回调函数callback，用端口号新建一个Socket，定义其readHandler功能为：只要从socket正常获取来自remoteEndpoint的数据包，就把它交给回调函数callback，过滤掉所有其他来源的数据包。

随后调用startRecv，开始接收并处理数据包。

void addRosRecv(PORT port, std::function<void(ROSMSG &)> callback)

传入一个本机端口号和回调函数callback，用端口号新建一个Socket，定义其readHandler功能为：把从socket获取的数据包翻译成ROS message（**但是好像不翻译也能用？**），交给回调函数callback。

随后调用startRecv，开始接收并处理数据包。

发送数据包：

void createSendSocket(const std::string &addr, const PORT &port)

传入目标主机的ip地址addr和端口号port，封装成一个broadcastEndpoint对象。新建一个socket。通过std::map，一个(addr, port)可以唯一对应到一个socket。令这个socket与broadcastEndpoint建立连接。

void DTransmit::sendBuffer(boost::asio::ip::udp::socket *socket, const void *buffer, std::size_t size)

传入createSendSocket创建的socket，数据缓冲区地址，数据包大小，用这个socket把数据包发过去。

void sendRaw(PORT port, const void *buffer, std::size_t size)

传入端口号，数据缓冲区地址，数据包大小。调用broadcast_addresses_获取数据要发到的所有ip地址，调用createSendSocket创建每个地址对应的socket（端口号都是传入的port），调用sendBuffer把数据包发到这些地址。

void DTransmit::sendRos(PORT port, ROSMSG &rosmg)

传入目标主机端口号和rosmg，将rosmg序列化（变为可以用socket发出的格式，**但是好像不变也能发？**），放入buffer中，调用sendRaw发出去。

dprocess

Network module 的“进程”类，Team和GameController的父类，负责新建一个线程，线程中循环调用Team和GameController的发包函数。

循环频率、是否采用SCHED_FIFO的线程调度策略均可修改。

dnetwork

所有机器人用于队内通信的端口号均为57335，定义在
workspaces\core\src\dconfig\include\dconfig\dconstant.hpp

所有机器人用于和裁判盒通信的端口号均为3838，裁判盒用于和机器人通信的端口号为3939，定义在 `workspaces\core\src\dnetwork\include\dnetwork\RoboCup\RoboCupGameControlData.h`

Team

本机内ROS通信

订阅vision，motion，behavior，gamecontroller，向/dbehavior_n/BehaviorInfo（n为本机编号）发布消息。

收消息

调用DTransmit的addRawRecv方法，回调函数为：

先根据条件筛选：

- 1) 数据包的大小与TeamInfo一致
- 2) 是本队的消息
- 3) 不是自己发出的消息

如果都满足，则向/dbehavior_n/TeamInfo发布。

发消息

通过tick()方法实现。

从vision，motion，behavior，gamecontroller获取信息后更新在info_中，调用DTransmit的sendRaw方法把info_发给局域网内所有主机的57335端口。

GameController

本机内ROS通信

没有订阅。

通过tick()方法（和Team中的tick不同），向/dbehavior_n/GCInfo（n为本机编号）发布消息，消息内容为info_。

收消息

调用DTransmit的addRawRecvFiltered方法，回调函数为：筛选掉不来自裁判盒的数据包和大小不符合RoboCupGameControlData数据类型的数据包，正常的数据包交给ParseData方法处理。

通过ParseData和其他方法，逐步筛选出发给本队（**有没有精确到本机？**）的数据包，如果相比之前一个有更新，则更新在info_中。

发消息

调用DTransmit的sendRaw方法，把ret_消息（包含队伍编号、机器人编号、表明自己在线的标记）发给发给局域网内所有主机的3939端口（但只有裁判盒会处理）。