GAPS接入

2024-03-02 01:03

目录

优点和缺点

优点：不用自己实现各种协议的发包和收包，自己开发业务层的body和head填充就行。

缺点：
1. 目前下载需要先收费。
2. 大部分内部代码做了实现屏蔽，没有指导都不太好调试。
3. 单进程的能力发压能力较弱，在ZPLAN项目中tgcp协议中的可靠udp发包，1000个机器人就会出现瓶颈。需要额外扩展多进程控制。
4. lua脚本为魔改版本，需要按照他的自定义格式来进行编写，不是标准的lua语言格式。每次更改后最好运行一遍./checklua进行检验。
5. 各种参数设置和可能的坑点没有详细的文档，而且该工具年代较老了，支持度不够。

协议支持

按照官网的简介支持http协议、tcp协议、tgcp协议、udp协议、websocket协议支持。数据压缩格式支持tdr协议解包发包。

在zplan接入过程中，因缺省部分能力，接入了protobuf数据格式解析和压缩能力，jsoncpp库来组http的json请求body。

接入步骤流程

按zplan项目为例，下载好demo文件后。跟开发对齐协议格式，使用的传输方式是啥，具体的协议字段和请求。然后再plugin中编写组包和组头文件。

- 发包流程主要在xxxscriptop.h中，主要是组请求body。然后调用`m_poPack`来打包，此时需要传入待打包buff指针和buffLen。

- 可以更改scriptop.h里面的声明以及更改实现，比如增加一个`m_pOpBufLen`变量来记录组包完成后的buff长度。在xxxpack.cpp中，实现head的pack和unpack。参数中`void *pItemsToPack, int nItemLen, char *pPackPkg, int PkgBufLen`分别为待打包内存数据，该数据块长度，打包好的内存块地址，打包缓冲区长度，其中`PkgBufLen`没啥用。PackMsg的返回值，该值传给`m_nPackedPkgLen`这个才是最后塞到不同协议里面的body_len长度。至此我们的body算打好了，`m_pPkgBuf`会将PackMsg塞给不同的协议他们再去打对应的协议头，比如http中填充的`Content-Type`、`Content-Length`；比如tconnd里面还会用tdr给压缩一下，在前面加上自己的头（对方地址和来源地址什么的）。

- xxxpkgprocess.h里面是对回包的处理，相当于先注册个回调函数，当有包回来时候会调对应的ProtocolID的函数。所以先要实现`StoreProtocolID`这个函数是告诉应该用注册map里面哪个回调的，主要作用是产生ProtocolID。但是注意是该函数是在unPackMsg之后才会运行到的，在解包头的时候做的一些操作会产出一个解完的`pUnpackPkg`这缓冲区内必须有协议ID信息因为在unpack是无法保存协议ID的，他是在`TPkgProcess`定义的，目前没有找到可以引用的方法。

- 然后再在bin/script中编写lua脚本，它这个lua只能按step1~N的步骤来进行运行，但是给了一些流程控制的方法，比如StayInStep、SetNextStep、GotoStep（没有暴露出来，查多个demo才知道……）。lua方法会在`taskFuncApi.lua`暴露，应该是c++实现的调用接口。lua脚本中的关键步骤是创立链接——`StartRobotConnectEx`、执

行在`xxxscriptop.h`注册的命令——`DoCommandEx`、判断执行结果——`GetRobotValue`。在启动一个事务时StartTrans，在判断结束一个事务时EndTrans，这样可以自动在报告平台生成对应事务的响应数据。

- 最后在robotdata/xxgame.xml中编写机器人的config，比如

```
<property name="stRobot" type="table" value="机器人配置">
        <property name="nStartUin" type="long" value="1000250" des="机器人起始qq账号"/>
        <property name="nTotalNum" type="long" value="1" des="机器人总数"/>
        <property name="nAccountFlag" type="long" value="0" des="账号类型：连续账号填0，不连续账号填1"/>
        <property name="szAccountFile" type="string" value="//bin/script/gamename/testdata/QQList.txt" des="账号文件，配置测试用qq账号，连续账号不需配置"/>
        <property name="nInitFlag" type="long" value="0" des="机器人初始化标识，是否需要在机器人身上设置初始化属性，0不需要，1需要"/>
        <property name="szInitAttrFile" type="string" value="QQInitAttr.txt" des="机器人初始化属性数据文件"/>
    </property>
```

这些配置会在test_init.lua、test_case.lua中读取并挂载在机器人上，`BatchOpt(g_nStartUin, g_nTotalNum, nPerLoopNum, nPerLoopWait, szScriptName);`这里读取的就是你在配置中填写的`nStartUin`、`nTotalNum`等信息。值得说一下的是他的整体循环为每个ScriptStep会分成多个批次挂载执行，这个取决你设置的nPerSecNum，当一个ScriptStep全部执行完之后进行下一个Step。当最后一个ScriptStep执行完时候再判断nCircleCnt循环执行次数是否完成。

经验和坑点

1. 好用的proto协议编译脚本
proto转c++头文件和cc

```
#!/bin/bash
protofolder="./protocol"
protocoldir=$(ls -l $protofolder |awk '/^d/ {print $NF}')

for dir in ${protocoldir}
do
    echo "protoc -I ${protofolder} --proto_path=${protofolder}/${dir} --cpp_out=${protofolder} ${protofolder}/${dir}/*.proto"
    protoc -I $protofolder --proto_path=${protofolder}/${dir} --cpp_out=${protofolder} ${protofolder}/${dir}/*.proto
done
```

将多个proto协议合并为一个.a库

```
CC              = gcc
CXX             = g++
AR              = ar rc

###编译标志###
C_FLAGS         =  -g -fPIC -std=c++11

OBJ_DIR = ./obj/
TARGET_DIR = ../../lib/
TARGET= $(TARGET_DIR)/libproto.a

RECURSION_SRC_DIR = $(shell find $(wildcard $(SRC_DIR)) -type d | grep -v '.svn' | grep -v 'obj'| grep -v 'lib')
SOURCE = $(foreach d, $(RECURSION_SRC_DIR), $(wildcard $(d)/*.cpp $(d)/*.c $(d)/*.C $(d)/*.cxx $(d)/*.cc))

SRC=$(notdir $(SOURCE))
SRCOBJ=$(addsuffix .o, $(basename $(SOURCE)))
OBJS=$(addprefix $(OBJ_DIR), $(SRCOBJ))

INC+=-I./protocol -I../../include
LIB = $(LIB_PATH) -lprotobuf

all:$(OBJS)
.PHONY:clean

$(OBJ_DIR)%.o: %.cpp
    @if [ ! -d $(OBJ_DIR) ];then mkdir -p $(OBJ_DIR); echo "mkdir -p $(OBJ_DIR)";fi
    $(CXX) $(INC) $(CXX_FLAGS) -c -g $< -o $@
    @echo compile   $@
$(OBJ_DIR)%.o: %.c
    $(CC) $(INC) $(C_FLAGS) -c -g $< -o $@
    @echo compile   $@
$(OBJ_DIR)%.o: %.cc
    $(CC) $(INC) $(C_FLAGS) -c -g $< -o $@
    @echo compile   $@

$(TARGET): $(OBJS)
    $(AR) $@ $^
    @echo  @$(AR) $@ $^
all: $(TARGET)

clean:
    @rm -f $(TARGET) $(OBJS)
    @echo rm $(OBJS)  $(TARGET)
```

2. gaps的apollo版本
apollo_lwip_version：20150805
apollo_version：1_1_14_102034
当时查问题需要对应的tconnd版本，但是平台方不开放接口给开发者。这里直接拷贝了新版本的一份Apollo头文件，在./lib里面的.so文件找到对应函数，命令是`nm ./lib/libapollo.so`。这种方法只能在函数调用关系和文件地址没变更的时候使用。

3. gaps的多进程控制
在脚本lua中写基于uin的控制流，再用shell依次进行启动多个gaps进程。下面是begin的脚本，end类似。
begin_all.sh

```
robot_num=2000
robot_batch=2
robot_config="testgame.xml"

for i in $(seq 0 $((robot_batch-1)))
do
    cp -r Zplan_Test Zplan_Test_${i}

    line=`sed -n '/nStartUin/='
Zplan_Test_${i}/bin/script/tgcp_demo/robotdata/${robot_config}`
    sed -i "/nStartUin/d" Zplan_Test_${i}/bin/script/tgcp_demo/robotdata/${robot_config}
    sed -i "$line i <property name=\"nStartUin\" type=\"long\"
value=\"$[$[$[robot_num/robot_batch]*i]+1000000]\" des=\"机器人起始qq账号\"/>"
Zplan_Test_${i}/bin/script/tgcp_demo/robotdata/${robot_config}

    line=`sed -n '/nTotalNum/='
Zplan_Test_${i}/bin/script/tgcp_demo/robotdata/${robot_config}`
    sed -i "/nTotalNum/d" Zplan_Test_${i}/bin/script/tgcp_demo/robotdata/${robot_config}
    sed -i "$line i <property name=\"nTotalNum\" type=\"long\"
value=\"$[robot_num/robot_batch]\" des=\"机器人总数\"/>"
Zplan_Test_${i}/bin/script/tgcp_demo/robotdata/${robot_config}

    cur_dir=$(pwd)
    cd Zplan_Test_${i}/bin
    sh begin.sh
    cd $cur_dir
done
```

4. lua机器人脚本的数字转str
`"Test"..uin`就行，不要使用base方法什么to_string什么的，会报错

5. 批量GBK转UTF-8
下载下来的文档和代码都是GBK的，在IDE上面看着十分费力，可以用下面的脚本进行转换。使用方法`./iconvfa.sh -R dirname`

```
#!/bin/env bash

function show_help
{
    echo "Usage:"
    echo "  iconvfa.sh [option] [file|dir]"
    echo -e "  from GB2312 to UTF-8, the old file will be replaced by the new converted file\n"
    echo "Options:"
    echo "  -R: convert files recursively, the following parameter should be the directory name"
}

# param 1: directory name
function convert_rescursive()
{
   local dir_path=`echo $1 | sed 's/\(.*\)\/$/\1/g'`
   local dir_names=`ls ${dir_path} -l | awk '/^d/{print $NF}'`

   # convert files in this directory
   local file_names=`ls ${dir_path} -l | awk '/^-/{print $NF}'`
   for file in ${file_names}
   do
       iconv -f ${from_code} -t ${to_code} ${dir_path}/${file} &amp;> /dev/null
       if [ $? == 0 ]; then
           iconv -f ${from_code} -t ${to_code} < ${dir_path}/${file} > $@.$$$$
           cp $@.$$$$ ${dir_path}/${file}
           rm -f $@.$$$$
           echo "File ${dir_path}/${file} is formatted."
       fi
   done

   # if the directory has no other directory, return 0
   if [ "${dir_names}X" == "X" ]; then
       return 0
   fi

   # continue convert files in directories recursively
   for dir in ${dir_names}
   do
       convert_rescursive "${dir_path}/${dir}"
   done
}

# defines
from_code="GB2312"
```

```
45    to_code="UTF-8"
46
47    case "$1" in
48    "-R")
49        ls $2 &amp;> /dev/null
50        if [ $? != 0 -o "$2X" == "X" ]; then
51            echo "#### error: please check the directory name follow the '-R' option!"
52            exit 1
53        fi
54        convert_rescursive $2
55        ;;
56    "")
57        show_help
58        ;;
59    *)
60        iconv -f ${from_code} -t ${to_code} $1 &amp;> /dev/null
61        if [ $? == 0 ]; then
62            iconv -f ${from_code} -t ${to_code} < $1 > $@.$$$$
63            cp $@.$$$$ $1
64            rm -f $@.$$$$
65            echo "File $1 is formatted."
66        else
67            echo "Convert wrong!"
68        fi
69        ;;
70    esac
```

坑

记录下

1. lwip有长尾的延迟响应，可能存在发出包但是对方tconnd没有接受处理，这个还在排查中。
2. bcs上面的网络不可靠，IDC请求的网络质量明显差于DEV的内网实验环境。在外网实际压测前先排查网络链路问题，最好在相同机房上，中间有多层中转都可能产生延迟或者丢包。
3. SetRobotValue有可能失败，暂时不知道为啥，尤其在设置字符串格式时候。可以bit法转为int32类型或者int64来表示暂时规避。
4. 抓包可以排查是否包真的有发出去，gaps的内部日志没有提供完全，比如想看tconnd是否发出。`tcpdump -iany udp -w ./udp.cap` 再使用wireshark进行解析排查。
5. http请求时碰到必须经过中转服才能成功解析的情况，或者使用curl工具进行请求。但是tcpdump查看两次的请求包都是一样的，暂时不知道啥原因，后面有空排查。
6. dev环境中做压测没有报告生成，只能通过自己打记录点的方式来判断。这里分享个统计脚本，在发包前后进行记录打ErrLog。

```
#encoding=utf-8

import getopt
import sys
import re

class QeqTimeCollect(object):
  def __init__(self, req_name):
     self.req_name = req_name
     self.avg_time = 0
     self.req_num = 0
     self.max_time = 0
     self.sum_time = 0
     self.cost_time_list = []

if __name__ == '__main__':
     options, args = getopt.getopt(sys.argv[1:], "r:", ["all"])

     req_name = ""
     for option, value in options:
         if option in ("--all"):
             req_name = "all"
             break
         if option in ("-r"):
             req_name = value

     data = {}
     with open(sys.argv[1], 'r') as f:
         lines = f.readlines()
         for line in lines:
             cur_req_name = re.findall('\[failed\] (.*?):', line)
             if cur_req_name:
                 data_collect = data.setdefault(cur_req_name[0],
QeqTimeCollect(cur_req_name[0]))
             else:
                 continue

             intervals = re.findall('interval:\[(.*)\]', line)
             if intervals:
                 cos_time = int(intervals[0])
                 data_collect.cost_time_list.append(cos_time)
                 data_collect.sum_time +=cos_time
                 data_collect.max_time = max(data_collect.max_time, cos_time)
                 data_collect.req_num += 1

     # if req_name == "all":
```

```
46        for req_name, data_collect in data.items():
47            if data_collect.req_num > 0:
48                print("----------------------------")
49                print(req_name+" :")
50                lag_req = filter(lambda x:x > 1000, data_collect.cost_time_list)
51                print('total req_num: {}, avg_cos: {:.2f} ms, lag_req_num:
      {}'.format(data_collect.req_num,
      data_collect.sum_time/data_collect.req_num,len(lag_req)))
52                data_collect.cost_time_list.sort()
53                _t = int(data_collect.req_num*0.7)
54                seventy_time = data_collect.cost_time_list[_t]
55                _t = int(data_collect.req_num*0.9)
56                ninety_time = data_collect.cost_time_list[_t]
57                print('70% cost_time: {:.2f} ms, 90% cost_time: {:.2f}
      ms'.format(seventy_time, ninety_time))
58                print('max_cos_time: {} ms'.format(data_collect.max_time))
```