COMP3331/9331 计算机网络与应用 作业

2024年第二学期

目录

前言

1. 有一定的python编程基础，9021过了就足够。
2. 确保之前lab的作业都搞定了。会基本的客户端与服务器的通信。
3. 只要及格的同学，慎重做这个assignment，做一个报告，写点代码。混个2分就行。总共是20分
4. 和tutor的人品，关系很大，24T1(完全靠测试评分。)和23T3，
5. 主体思路：能通信的两个端点，传输的内容和协议有关。剩下的就是逻辑的内容了。
6. 我自己实现的：client，50行，server，150行。在历学期，算是代码量较小的学期任务了。
7. 一定要看完任务描述，自己认真看完。再听本节课

目标和学习目标

目标是：

1. 更深入地理解DNS的工作原理。
2. 练习使用UDP传输协议的套接字编程。
3. 设计一个应用层协议。

主文件

- 主文件（ `master.txt` ）包含文本格式的资源记录（RRs）。
- 主文件中的每一行包含一个资源记录，包括域名、类型和数据。
- 服务器启动时读取主文件以初始化内存缓存的资源记录。

```
RR = [["foo.example.com.","CNAME","bar.example.com."],
     ["foo.example.com.","CNAME","bar.example.com."],]
with open("") as f:
    RR
```

消息格式

法一：简单，也能拿高分的方法。

避免查重的方法：1.你的格式定义的顺序变化一下，2.你的分隔符变化一下-!,&^

```
message = "qid qtype qname"
message = "1 A www.baidu.com 192.168.3.1"
message = "1^A^www.baidu.com 192.168.3.1"
message = "Q^1^A^www.baidu.com 192.168.3.1"
message = "R^1^A^www.baidu.com 192.168.3.1"
sender发送:
client_socket.sendall(message.encode())

接收端接收:
received = client_socket.recv(1024).decode()
received = received.split(" ") # received = ["1","A","www.baidu.com","192.168.3.1"]
```

您必须设计客户端查询和服务器响应的消息格式，可能包括以下部分：

到底要设计成类的形式，还是字符串的形式？！个人建议，选择为字符串。

1. **头部**：包含查询ID（qid）。
2. **问题**：包含查询类型（qtype）和目标域名（qname）。
3. **答案**：包含回答查询的资源记录。
4. **授权**：包含指向权威名称服务器的资源记录。
5. **附加**：包含其他相关的资源记录。

法二：特别优美，特别完美，能拿满分，但难的方法，类和对象的好处，代码风格好，耦合性低。

```
class DNS:
    def __init__():
        ID = ""
        number = ""

在sender里，想要发送这个类:
pickle库（一些把python中各种数据结构，转为二进制结构的库都行）

在receiver里，把这个消息，通过pickle再转成对应的类。处理类中的信息。
```

法三：json，xml，jwt等等等等

图14-3　DNS查询和响应的一般格式

客户端

文件名和参数

- 主代码必须在 `client.c` 、 `Client.java` 或 `client.py` 中。
- 客户端接受以下参数：
  - `server_port` ：服务器监听的UDP端口号。
  - `qname` ：查询的目标域名。
  - `qtype` ：查询类型。
  - `timeout` ：等待响应的持续时间，超过此时间则视为失败。

sys库的使用。

```
python  client.py 50001 www.baidu.com A 1
```

```
import sys
port = int(sys.argv[1])
type = sys.argv[2]
if type == "A":
    xxxxx
```

客户端操作

1. 构造并发送查询消息到服务器。
2. 等待响应，直到达到指定的超时时间。
3. 打印响应或如果超时则打印错误消息。
4. 终止。

客户端输出

- 必须包含查询ID、问题部分、以及答案、授权和附加部分的资源记录。

服务器

文件名和参数

- 主代码必须在 `server.c` 、 `Server.java` 或 `server.py` 中。
- 服务器接受以下参数：
  - `server_port` ：服务器监听客户端查询的UDP端口号。在50000到60000之间。

```
python server.py 50001
```

服务器操作

1. 读取主文件并初始化内存缓存。
2. 绑定到回环接口的指定端口。
3. 监听客户端查询并并发处理，随机延迟0到4秒。
4. 使用缓存响应查询。

查询处理

1. 匹配查询中的qname到资源记录。
2. 处理CNAME、引用和直接匹配。
3. 发送适当的响应给客户端。

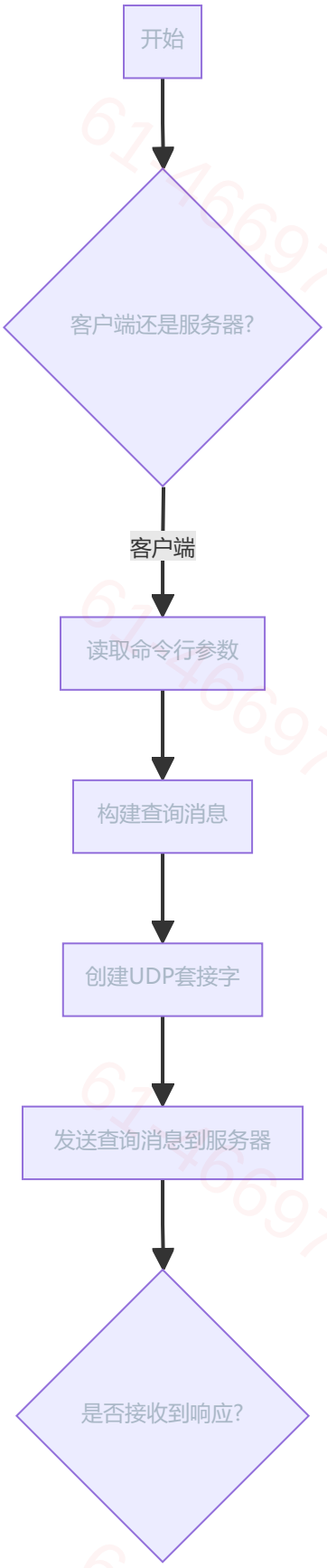
服务器输出

- 必须记录发送和接收的消息，包括时间戳、客户端端口号、查询ID、查询名称、查询类型和处理延迟。

报告

- 提交一份不超过2页的报告，命名为 `report.pdf` ，包括：
- 格式不算分
  - 使用的编程语言和代码组织的详细信息。
  - 程序设计和数据结构（dns设计的那个字符串）。

- ~~已知的限制。~~
- ~~借用代码的来源和修改。~~
- 推荐用processon，或者draw.io画两个流程图。
- 可以加一些思考内容：tradeoff，权衡。
- 取巧方法：把你最后写好的代码，喂给chatgpt，生成mermaid格式的流程图。



CPU-juicy

附录

语言和允许的功能

- 可以使用C、Java或Python。
- 代码必须能在CSE服务器上编译和运行。
- 自己实现所有功能，不得使用现成的库或工具。

额外说明

- 独立完成作业，不允许小组合作。
- 经常备份您的工作。
- 彻底测试您的代码。
- 使用课程论坛提问和讨论（不要共享代码）。

提交

- 使用规定的文件名提交报告和代码文件。
- 确保您的提交被系统接受。

迟交政策

- 每迟交一天，扣除5%的成绩，最多5天。

请定期检查课程网站的更新和澄清信息。

客户端伪代码

如果lab2，和lab3是自己写的，并且这次作业，没给其他人看，可以直接复制粘贴。

```
# 客户端伪代码

## 导入必要的库

## 构建查询消息的函数
函数 :
   生成随机的查询ID (qid)
   构建查询消息 (query)
   返回编码后的查询消息和查询ID

## 解析响应消息的函数
函数 :
   解码响应消息
   打印响应内容

## 客户端主函数
函数  :
```

```
    设置服务器地址（"localhost" 和给定的端口）
    创建UDP套接字
    设置套接字超时时间

    尝试：
        构建查询消息
        发送查询消息到服务器

        接收服务器响应
        解析并打印响应消息
    如果超时：
        打印 'timed out'
    最终：
        关闭套接字

## 脚本入口
如果参数数量不正确：
    打印用法说明并退出

获取命令行参数：

调用客户端主函数：

```

服务器端伪代码

1．多线程：

2．逻辑上：从列表里怎么就能拿出来信息？递归式地拿信息，5.3

```
# 服务器端伪代码

## 导入必要的库


## 加载主文件的函数
函数 ：
    初始化记录字典（records）
    打开并读取文件
    对于文件中的每一行：
        分割行内容为域名、记录类型和数据
        如果记录字典中没有该域名：
            初始化该域名的记录列表
        将记录类型和数据添加到该域名的记录列表
    返回记录字典，推荐是字典。RR是那种特别典型的key-value的数据

## 构建响应消息的函数，这个可能是最复杂的内容了。
函数 ：
    分割查询消息为查询ID、域名和查询类型
    初始化响应消息，“x-x-x-x”
```

```
    初始化答案、授权和附加部分 空内容

     查找给定域名的A记录
    构建响应逻辑（省略具体实现细节）
    while True:
          直到匹配上需要的答案，或者压根没这个答案就退出。
          if qname 在这个大字典里:
               这个找到的key-value是否是需要的qtyme类型？ if
               if是:
               加到响应消息里
               if not:
                    if 是cname，那就修改qname，重新查 :


                    else :
                         是否退出? contine

          else: 不在
               发空消息的逻辑,没有找到答案的消息内容。

          最后添加，权威信息，additional信息等等。

    返回编码后的响应消息

## 日志记录函数
函数 :每次，server有发送接收的动作，给这个函数传一些参数，
    获取当前时间戳，date,time,datetime
    构建日志消息
    如果是接收方向 ("rcv"):
       添加延迟信息
    打印日志消息  print( )

## 处理查询的函数
函数 :传过去
    分割查询消息为查询ID、域名和查询类型
    获取客户端端口号
    随机生成处理延迟时间
    记录接收查询的日志
    延迟处理
    构建并发送响应消息，发给socket
    记录发送响应的日志

## 服务器主函数
函数 :
    设置服务器地址（localhost 和给定的端口）
    创建UDP套接字并绑定到地址

    加载 master.txt 记录
    打印服务器监听信息

    无限循环:
       接收客户端查询，recv函数
```

```
    解码查询消息
    创建线程处理查询

## 脚本入口
如果参数数量不正确:
  打印用法说明并退出

获取命令行参数: `server_port`

调用服务器主函数: `server(server_port)`
```

threading库，多线程

```python
import threading
import time

# 定义线程要执行的任务函数
def print_numbers():
    for i in range(1, 6):
        print(f"Number: {i}")
        time.sleep(0.1)

def print_letters():
    for letter in 'ABCDE':
        print(f"Letter: {letter}")
        time.sleep(0.1)

# 创建两个线程
thread1 = threading.Thread(target=print_numbers)
thread2 = threading.Thread(target=print_letters)

# 启动线程
thread1.start()
thread2.start()

# 等待所有线程完成
thread1.join()
thread2.join()

print("All threads have finished execution.")
```

多线程和处理消息的结合

```python
import threading
import time

# 定义线程要执行的任务函数
```

```
def process_query():
    for i in range(1, 6):
        print(f"Number: {i}")
        time.sleep(0.1)

在主函数里:
    while True:
        50001, 监听
        来了一个新的查询
       threading.Thread(target=process_query,args = (socktet,xxxx)).start()
```


CPU-juicy

报告

Used Language: Python3

Segment : ACK SYN FIN DATA RESET
Action: Send Segment Receive Segment
Object: sender receiver

To do a classfication, it will be clear through all the way.

Organisation: receiver.py sender.py utility.py const.py

receiver.py and sender.py are the files which run, utility.py provide function like creating segment and creating segment, const.py provide constant numbers.

Program Design

Just use class for sequence number while other operations are finnished in oriented procession.It will go fluently.With using sleep function for the maximum 3 resend time, it doesn't need timer which it will be used often if with OOP.And use two function in main program, one is "rece" and the other is "send".All the left are helper function.
In summary, one class and two funcion, form a forward straigh and readable program.

Data Structure

In sender side:

```
Use 'result' key to record if has received or not.
```

```
Use 'send_index' to record the seq.
Use  'time_stamp' to record exact time.
Use 'data_len' to record length of data.
Use  'ack_received' to record how many times the ack has been received
```

In receiver side:

```
Just set a dict to reocrd the sequence and meassgae that will be sorted
and writed into file.
```

Operation

It is the route how the data will generate from left to right.

And some dict or map-relation is at the bottom

Name	**Source**					
max_win	input					
rto	input					
receiver_port	input--+IP-->	receiver_addr				
FileToSend	input-->	FileToSend →	send_all_message--> MSS--⟶	pkts →	N = len(pkts) →	window_data
sender_log	input					
sender	Create socket Object					
sender_start_time	Exactly time.time()					
summary_dict	log final part					
ISN	random					
lef, rig =0 ,0	Windows Tag					

summary_dict			
'Amout_Data_No_Repeat'	'Data_Seg_Received_No_Repeat'	'Data_Seg_Resent_Repeat'	'ACK_Seg_Received_Repeat'

window_data				
'result'	'send_index'	'time_stamp'	'data_len':	'ack_received'
T F	0			0
DATA_NUM	ACK_NUM	SYN_NUM	FIN_NUM	RESET_NUM
0	1	2	3	4

Trades_offs

Using thread can not be ignore in sender side .However , I don not use thread in receiver because there is no necessity to send ACK in delay which is said in assignment.But it scasifice some accuracy in time.