中山大學
SUN YAT-SEN UNIVERSITY

本科生实验报告

学生姓名：　丁晓琪

学生学号：　22336057

专业名称：　计科

一：实验任务

使用protobuf和gRPC或者其他RPC框架实现消息订 阅（publish-subscribe）系统，该订阅系统能够实现简单的消 息传输，还可以控制消息在服务器端存储的时间。

二：实验思路

定义服务`SubService`：

远程调用的订阅请求函数`Subscribe`，接收要定义的主题名和客户端的订阅id

服务器流式RPC方法`Message_Stream`

各种消息类型

- 创建 gRPC 客户端和服务器的类和方法
- 服务器定义：
  - 完成`Subsribe`订阅函数的定义：注册订阅号和订阅话题
  - 完成`Message_Stream`流式传输函数定义：根据传进订阅号找到订阅话题，检测话题消息的存储时间是否超过限制且进行清理，将话题的最新消息打包发送给客户端。连接未断开时，只要话题消息更新就发送给客户端
  - 定义两个话题`time`,`count`，并且创建两个线程用于自动更新话题内消息
- 客户端定义：
  - `client`：订阅`time`话题，打印收到的消息
  - `client`：订阅`count`话题，打印收到的消息

三：实验过程：

[1].

- 定义服务`SubService`
    - 原理：Protobuf是一种结构化数据存储格式，用于结构化数据串行化，适合做数据存储或者RPC数据交换格式。要将`message`和RPC一起使用时，可以在`.proto`文件中定义RPC服务接口，`protobuf`编译器会根据选择语言生成RPC接口代码
    - `message`定义：
        - 订阅请求`SubscrobeRequest`：

```
message SubscribeRequest{
    string topic_name=1;#订阅主题名，字段编号为1
    string id=2; #申请订阅的客户端id，字段编号为2
}
```

        - 订阅标志（请求订阅的响应）`SubscribeResponse`：

```
message SubscribeResponse{
    int32 flag=1; //表示是否订阅成功
}
```

        - 流式传输中客户端需要传入的信息`request_id`：

```
message request_id{
    string id=1;//传入的客户id
}
```

        - 流式传输中服务器返回的话题最新消息`topic_Message`：

```
message topic_Message{
    string topic=1;//订阅的主题
    string content=2;//传递内容
    int64 timestamp=3; //时间戳
}
```

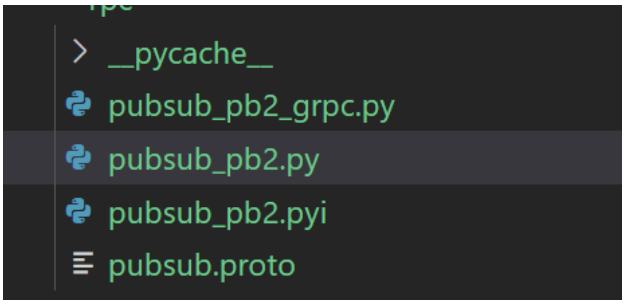
    - 定义rpc服务接口`SubService`：内含两个RPC方法

```
service SubService{ //gRPC服务
 //订阅主题的rpc方法：客户端发起订阅请求，服务器返回订阅是否成功标志
 rpc Subscribe(SubscribeRequest) returns(SubscribeResponse);
 //流式推送消息：客户端订阅成功后，流式服务时传入客户id，服务器将对应订阅主题内消息推送，是Message的信息流
 rpc Message_Stream(request_id) returns (stream topic_Message);
}
```

    - 编译`.protoc`文件。生成相应的`python`代码：

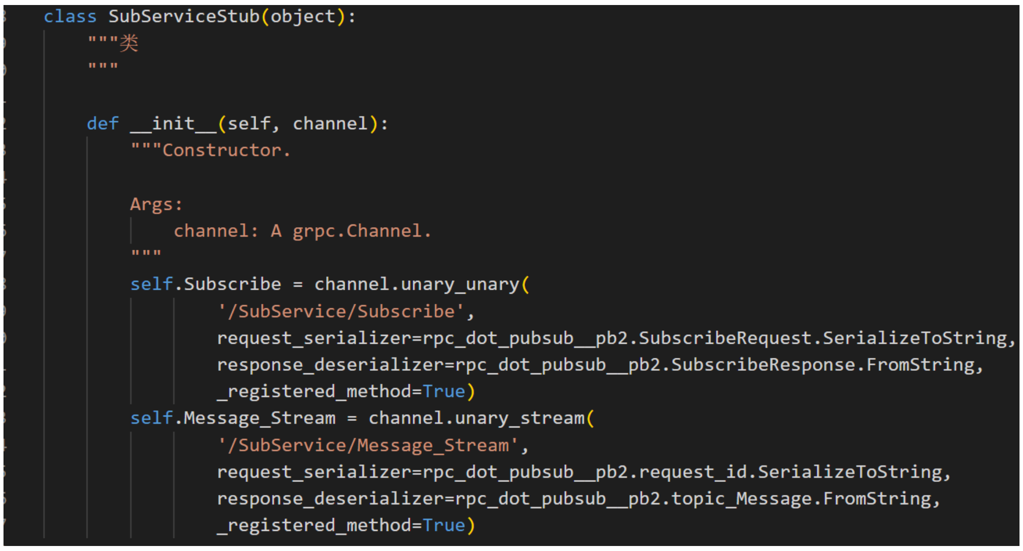
```
 python -m  grpc_tools.protoc //以模块形式运行grpc_toolss.protoc(能够生成
gRPC服务和消息类的python代码)
 -I ./    //在当前目录查找导入的.protoc文件
 --python_out=. //生成的消息类输出目录在当前目录
 --pyi_out=.
 --grpc_python_out=.  //生成的grpc相关的文件（服务和存根）放在
当前目录
rpc/pubsub.proto
```

- 生成结果：



```
> __pycache__
pubsub_pb2_grpc.py
pubsub_pb2.py
pubsub_pb2.pyi
pubsub.proto
```

- 服务和存根代码在`pubsub_pb2_grpc.py`文件中
  - 客户端存根：

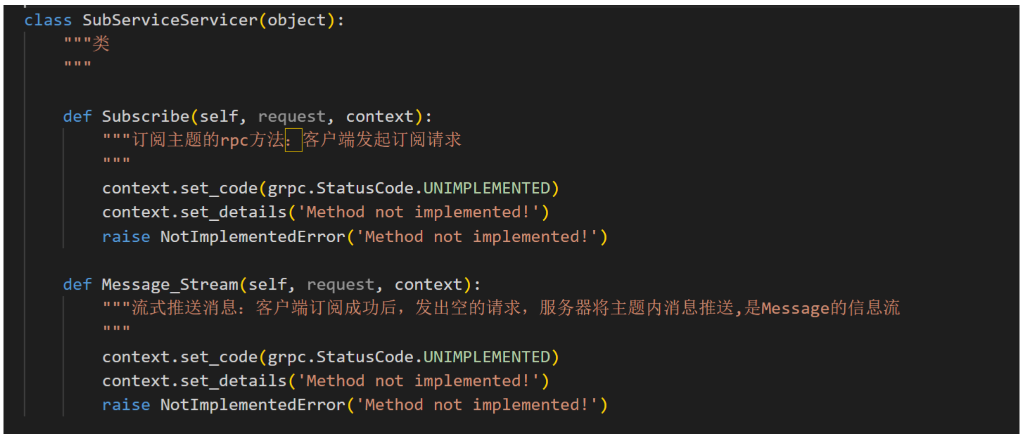


```python
class SubServiceStub(object):
    """类
    """

    def __init__(self, channel):
        """Constructor.

        Args:
            channel: A grpc.Channel.
        """
        self.Subscribe = channel.unary_unary(
                '/SubService/Subscribe',
                request_serializer=rpc_dot_pubsub__pb2.SubscribeRequest.SerializeToString,
                response_deserializer=rpc_dot_pubsub__pb2.SubscribeResponse.FromString,
                _registered_method=True)
        self.Message_Stream = channel.unary_stream(
                '/SubService/Message_Stream',
                request_serializer=rpc_dot_pubsub__pb2.request_id.SerializeToString,
                response_deserializer=rpc_dot_pubsub__pb2.topic_Message.FromString,
                _registered_method=True)
```

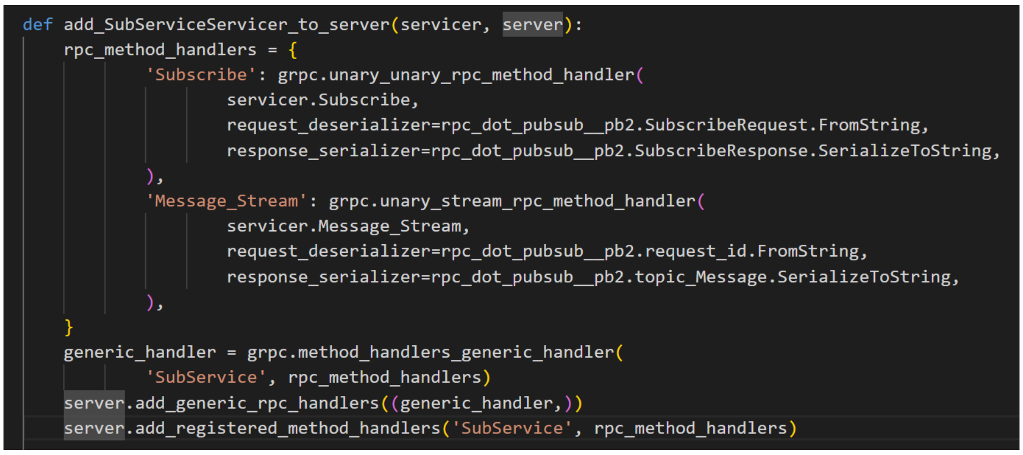
  - 服务器服务类：



```python
class SubServiceServicer(object):
    """类
    """

    def Subscribe(self, request, context):
        """订阅主题的rpc方法:客户端发起订阅请求
        """
        context.set_code(grpc.StatusCode.UNIMPLEMENTED)
        context.set_details('Method not implemented!')
        raise NotImplementedError('Method not implemented!')

    def Message_Stream(self, request, context):
        """流式推送消息: 客户端订阅成功后, 发出空的请求, 服务器将主题内消息推送,是Message的信息流
        """
        context.set_code(grpc.StatusCode.UNIMPLEMENTED)
        context.set_details('Method not implemented!')
        raise NotImplementedError('Method not implemented!')
```



```python
def add_SubServiceServicer_to_server(servicer, server):
    rpc_method_handlers = {
            'Subscribe': grpc.unary_unary_rpc_method_handler(
                    servicer.Subscribe,
                    request_deserializer=rpc_dot_pubsub__pb2.SubscribeRequest.FromString,
                    response_serializer=rpc_dot_pubsub__pb2.SubscribeResponse.SerializeToString,
            ),
            'Message_Stream': grpc.unary_stream_rpc_method_handler(
                    servicer.Message_Stream,
                    request_deserializer=rpc_dot_pubsub__pb2.request_id.FromString,
                    response_serializer=rpc_dot_pubsub__pb2.topic_Message.SerializeToString,
            ),
    }
    generic_handler = grpc.method_handlers_generic_handler(
            'SubService', rpc_method_handlers)
    server.add_generic_rpc_handlers((generic_handler,))
    server.add_registered_method_handlers('SubService', rpc_method_handlers)
```

[2].

- 服务器端：
    - 实现`SubService_Server`类：继承自动生成的服务器代码`SubServiceServicer`类，补充`Subscribe`和`Message_Stream`的实现：
        - 类成员：

```python
    def __init__(self) -> None:
        self.subscribes_topic={}  #客户订阅表，键值对为 (客户id，订阅话题)
        self.message_queue={}     #话题消息表，键值对为 (订阅话题，消息队列)
        self.ttl=20               #消息存储时间，初始化为20s
        self.lock=threading.Lock() #消息队列同步锁，避免多个客户同时对一个话题的消息队列操作时带来的不一致

        self.message_queue["time"]=[] #(ts,mes) #初始化time话题，消息队列存储结构为（时间戳，消息）
        self.message_queue["count"]=[] #          初始化计数器话题，消息存储结构为（时间戳，消息）
        self.generator_lock=threading.Lock()     #后面要定义两个线程在话题中自动生成一些消息且打印相关消息，线程锁避免打印不完整被切换线程

        self.message_generator_thread_time = threading.Thread(target=self.generate_messages_time)
        self.message_generator_thread_time.daemon = True  # 确保线程在程序退出时自动终止
        self.message_generator_thread_time.start()   #创建定期生成time话题消息的线程

        self.message_generator_thread_count = threading.Thread(target=self.generate_messages_count)
        self.message_generator_thread_count.daemon = True  # 确保线程在程序退出时自动终止
        self.message_generator_thread_count.start()   #创建定期生成topic话题消息的线程
```

        - 自动生成消息的线程：

```python
    def generate_messages_time(self):
        # 每5秒向"time"主题生成一条消息，但是因为有锁，具体生成间隔可能比2大
        i=8
        while (i>0):
            with self.generator_lock:
                i-=1
                time.sleep(5)
                current_time = time.strftime("%Y-%m-%d %H:%M:%S", time.gmtime())
                message = f"Current time is {current_time}"  #创建消息
                timestamp = int(time.time())                 #以当前时间创建时间戳
                self.message_queue["time"].append((timestamp,message)) #加入time的消息队列
                print("add topic time:",message,timestamp)  #打印添加消息成功

    def generate_messages_count(self):
        #每5秒向"count"主题生成一条消息，但是因为有锁，具体生成间隔可能比8大
        i=20
        while (i>0):
            with self.generator_lock:
                time.sleep(8)
                current_count = 20-i
                message = f"Current count is {current_count}"  #创建消息
                timestamp = int(time.time())                   #以当前时间创建时间戳
                self.message_queue["count"].append((timestamp,message)) #加入count消息队列
                print("add topic count",message,timestamp)  #打印添加消息成功
                i-=1
```

- 实现在`pubsub.protoc`中定义的RPC的两个方法：
  - `Subscribe`：
    - 参数：`request`：类型为`SubscrobeRequest`，包含客户id和订阅话题

      `context`：上下文对象，包含元数据，截止时间，取消状态等等
    - 操作：查找客户订阅的话题，如果话题在话题消息注册表`message_queue`中就注册客户和其订阅的话题在`subscribes_topic`并且返回成功标志，否则返回失败标志
    - 返回：`SubscribeResponse`成功为1，失败为0

```
    def Subscribe(self,request,context):
        topic_name=request.topic_name
        flag=1
        if topic_name not in self.message_queue: #要订阅的话题没有注册
            flag=0 #返回订阅失败

        custom_id=request.id
        self.subscribes_topic[custom_id]=topic_name
        #返回
        return pubsub_pb2.SubscribeResponse(flag=flag)
```

- `Message_Stream`:
  - 作用：当客户注册成功后，要获得订阅消息，调用此方法在客户端的存根，传入客户id，服务器返回未过期的最新消息，且在连接期间，数据更新时会继续传输最新消息。只有发送话题消息时会消除话题的过期消息，避免频繁检查
  - 参数：`request_id`客户id
  - 返回：`topic_Message`话题最新消息

```
    def Message_Stream(self,request,context):
        #1. 提取客户端信息,查找订阅的信息（假设一次只订一个话题）
        custom_id=request.id
        topic_name=self.subscribes_topic[custom_id]
        read=-1  #记录客户端读取的话题消息当前位置的索引

        while(1):
        #2.去除过期消息
            with self.lock:
                current_time = int(time.time())
               # 创建一个新列表来存储过滤后的消息
                filtered_messages = []
               # 遍历 self.message_queue[topic_name] 中的每个元素
                for ts, msg in self.message_queue[topic_name]:
                # 检查消息是否未过期
                     if current_time - ts <= self.ttl:
                 # 如果未过期，则添加到 filtered_messages 列表中

                        filtered_messages.append((ts, msg))

                    else:
                        print("time out!",topic_name,"message pop!",msg)
                        read-=1 #删掉的肯定是read前面的消息，放在队列越前面，存放时间越久，越有可能过期被删掉，read指向的对象位置向前,read索引向前

            # 用过滤后的消息列表替换原来的列表
                self.message_queue[topic_name] = filtered_messages
       #3. 发送最新消息
            #记录当前订阅话题消息队列长度
            len_message=len(self.message_queue[topic_name])
            #发送消息:
            with self.lock:
                #只有read索引比消息队列最后索引小才发送消息，否则会重复给客户端发送消息，每次最新消息只发送一次
                if self.message_queue[topic_name] and read<(len_message-1):
                    read=len_message-1
                    timestamp,message=self.message_queue[topic_name][len_message-1]
                    #打包响应消息未topic_Message，并且返回一次消息
                    response=pubsub_pb2.topic_Message(topic=topic_name,content=message,timestamp=timestamp)
                    yield response

            #防止频繁检查消息队列消息是否过期和消息队列是否更新
            time.sleep(1)
```

- 设置和启动gRPC服务器:

```
def serve():
    port="40000" #设置服务器的监视端口为40000
    server=grpc.server(futures.ThreadPoolExecutor(max_workers=10))#实例化易感染具有10个工作线程的线程池执行器，则服务器可以同时处理多达 10 个并发请求
    pubsub_pb2_grpc.add_SubServiceServicer_to_server(SubService_Server(),server)#将SubService_server上面定义的服务类添加到gRPC服务器，为了处理定义的RPC调用
    server.add_insecure_port('[::]:' + port)  # 为服务器添加一个不安全的监听端口。[::] 表示服务器将监听所有可用的网络接口
    server.start() #启动服务器，使其开始监听请求
    print("Server started, listening on " + port)
    server.wait_for_termination() #使服务器保持运行状态，直到它被明确地关闭或终止
```

[3].

- 客户端1：订阅`time`主题消息

```
def run():
    #1. 创建和本机40000端口的gRPC通道
     with grpc.insecure_channel('localhost:40000') as channel:
          #2. 创建存根，调用自动生成代码里面的存根类
          stub=pubsub_pb2_grpc.SubServiceStub(channel)
          #3. 发送订阅请求
          request=pubsub_pb2.SubscribeRequest(topic_name="time",id="123")
          response=stub.Subscribe(request)
          if(response.flag==1):
              print("sub succeed")
          else:
              print("fail")
          #4.从流中获得消息并且打印消息
          request2=pubsub_pb2.request_id(id="123")
          for message in stub.Message_Stream(request2):
            print(f"Received message for topic '{message.topic}':")
            print(f"  Content: {message.content}")
            print(f"  Timestamp: {time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(message.timestamp / 1000))}")  # 将时间戳转换为可读格式
```

- 客户端2：订阅`count`主题消息

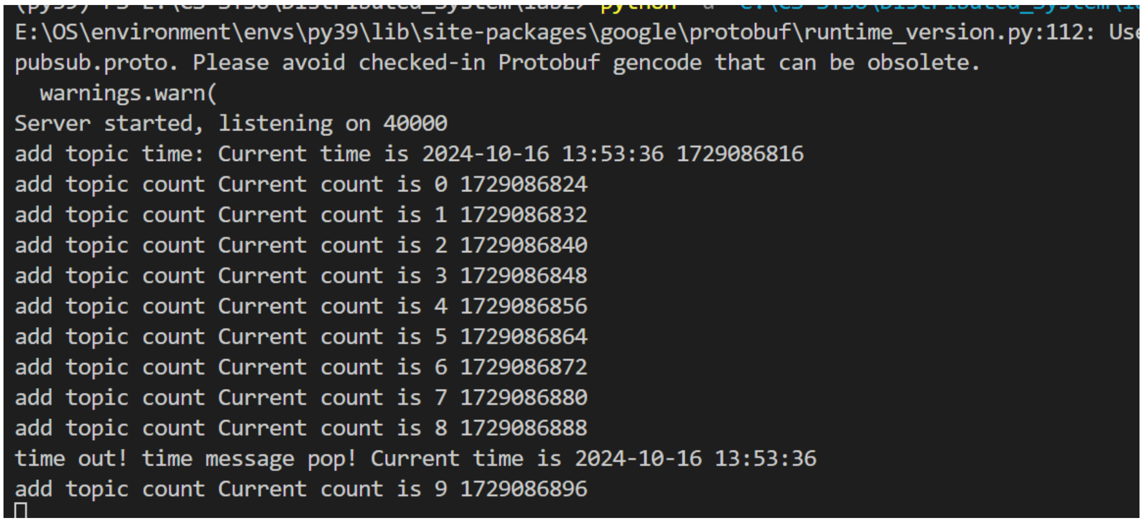
```
def run():
     with grpc.insecure_channel('localhost:40000') as channel:
          #创建存根
          stub=pubsub_pb2_grpc.SubServiceStub(channel)

 request=pubsub_pb2.SubscribeRequest(topic_name="count",id="456")
          response=stub.Subscribe(request)
          if(response.flag==1):
              print("sub succeed")
          else:
              print("fail")

          request2=pubsub_pb2.request_id(id="456")
          for message in stub.Message_Stream(request2):
            print(f"Received message for topic '{message.topic}':")
            print(f"  Content: {message.content}")
            print(f"  Timestamp: {time.strftime('%Y-%m-%d %H:%M:%S',
time.localtime(message.timestamp / 1000))}")  # 将时间戳转换为可读格式
```

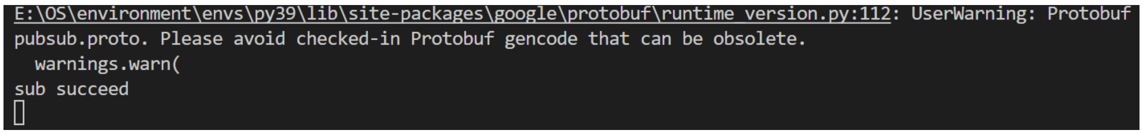
四：实验结果

- 检测消息订阅系统是否能够删除过期消息：
  先调整`time`自动生成线程，改成只生成一个消息，再开启服务端，生成time消息后等待一段时间（等待消息过期）再启动客户端订阅话题
  服务器只生成一个time消息，且在客户端订阅时检查是否过期，并且将其删除



```
E:\OS\environment\envs\py39\lib\site-packages\google\protobuf\runtime_version.py:112: Use
pubsub.proto. Please avoid checked-in Protobuf gencode that can be obsolete.
  warnings.warn(
Server started, listening on 40000
add topic time: Current time is 2024-10-16 13:53:36 1729086816
add topic count Current count is 0 1729086824
add topic count Current count is 1 1729086832
add topic count Current count is 2 1729086840
add topic count Current count is 3 1729086848
add topic count Current count is 4 1729086856
add topic count Current count is 5 1729086864
add topic count Current count is 6 1729086872
add topic count Current count is 7 1729086880
add topic count Current count is 8 1729086888
time out! time message pop! Current time is 2024-10-16 13:53:36
add topic count Current count is 9 1729086896
```

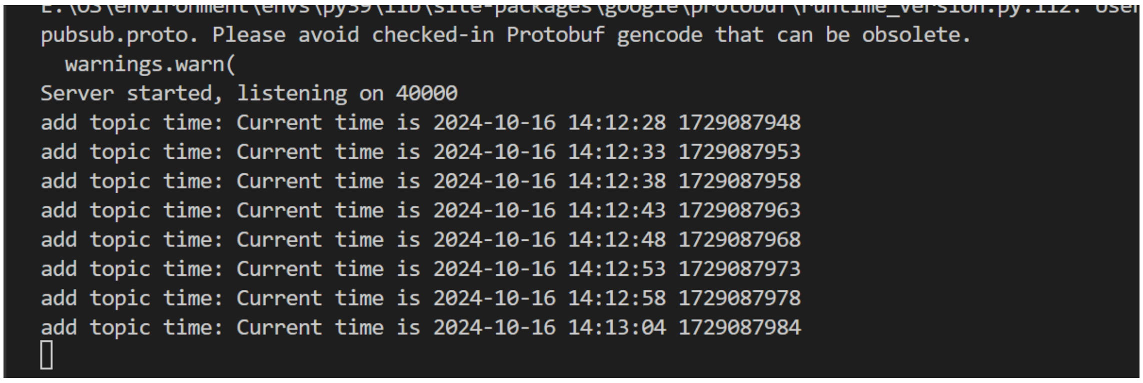
客户端没有收到任何消息：



```
E:\OS\environment\envs\py39\lib\site-packages\google\protobuf\runtime_version.py:112: UserWarning: Protobuf
pubsub.proto. Please avoid checked-in Protobuf gencode that can be obsolete.
  warnings.warn(
sub succeed
```

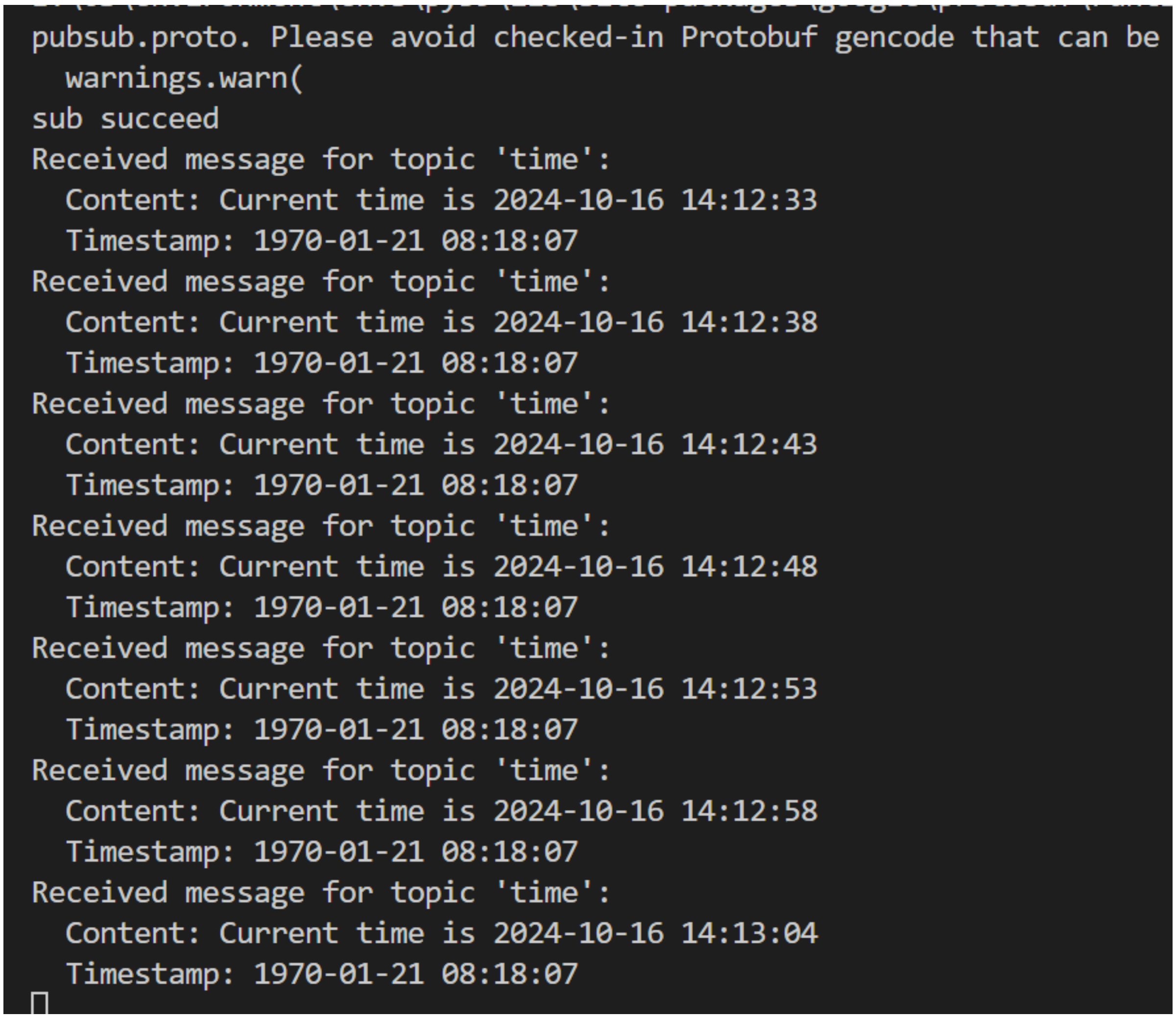
- 检验两个客户端同时订阅同一个话题：先开启服务器端，等待一段时间后开启客户端1，等待一段时间后，再在不同终端运行相同的客户端代码

  服务器端：



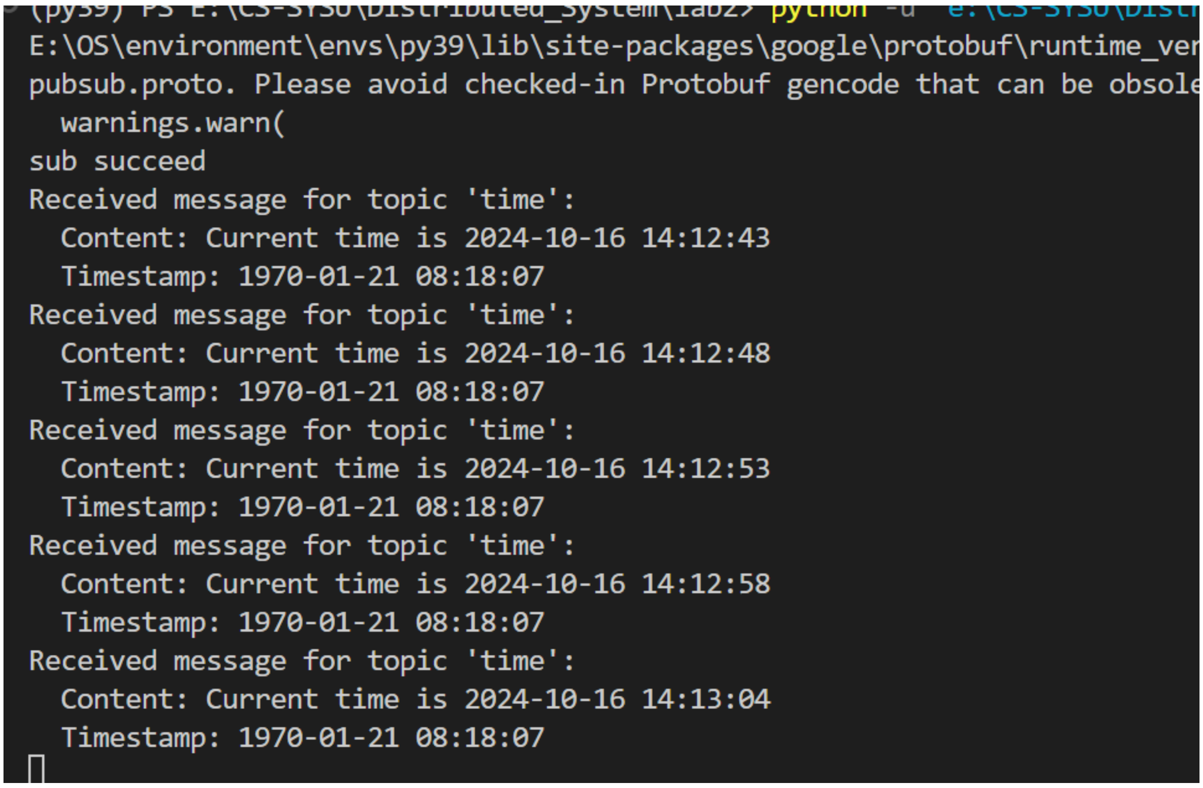
```
E:\OS\environment\envs\py39\lib\site-packages\google\protobuf\runtime_version.py:112: User
pubsub.proto. Please avoid checked-in Protobuf gencode that can be obsolete.
  warnings.warn(
Server started, listening on 40000
add topic time: Current time is 2024-10-16 14:12:28 1729087948
add topic time: Current time is 2024-10-16 14:12:33 1729087953
add topic time: Current time is 2024-10-16 14:12:38 1729087958
add topic time: Current time is 2024-10-16 14:12:43 1729087963
add topic time: Current time is 2024-10-16 14:12:48 1729087968
add topic time: Current time is 2024-10-16 14:12:53 1729087973
add topic time: Current time is 2024-10-16 14:12:58 1729087978
add topic time: Current time is 2024-10-16 14:13:04 1729087984
```

客户端1：获得自从订阅来的最新消息



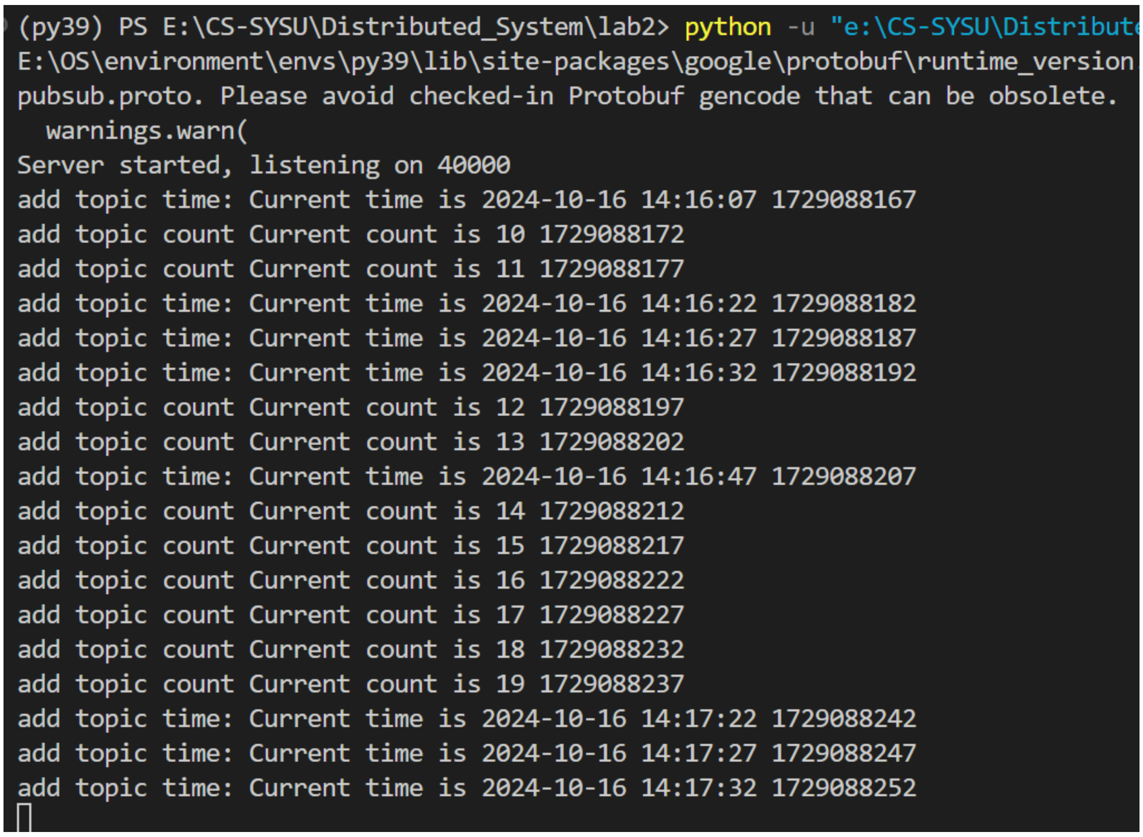
```
pubsub.proto. Please avoid checked-in Protobuf gencode that can be
  warnings.warn(
sub succeed
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:12:33
  Timestamp: 1970-01-21 08:18:07
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:12:38
  Timestamp: 1970-01-21 08:18:07
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:12:43
  Timestamp: 1970-01-21 08:18:07
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:12:48
  Timestamp: 1970-01-21 08:18:07
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:12:53
  Timestamp: 1970-01-21 08:18:07
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:12:58
  Timestamp: 1970-01-21 08:18:07
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:13:04
  Timestamp: 1970-01-21 08:18:07
```

客户端2：订阅的比客户端1晚，所以得到的消息更晚更少



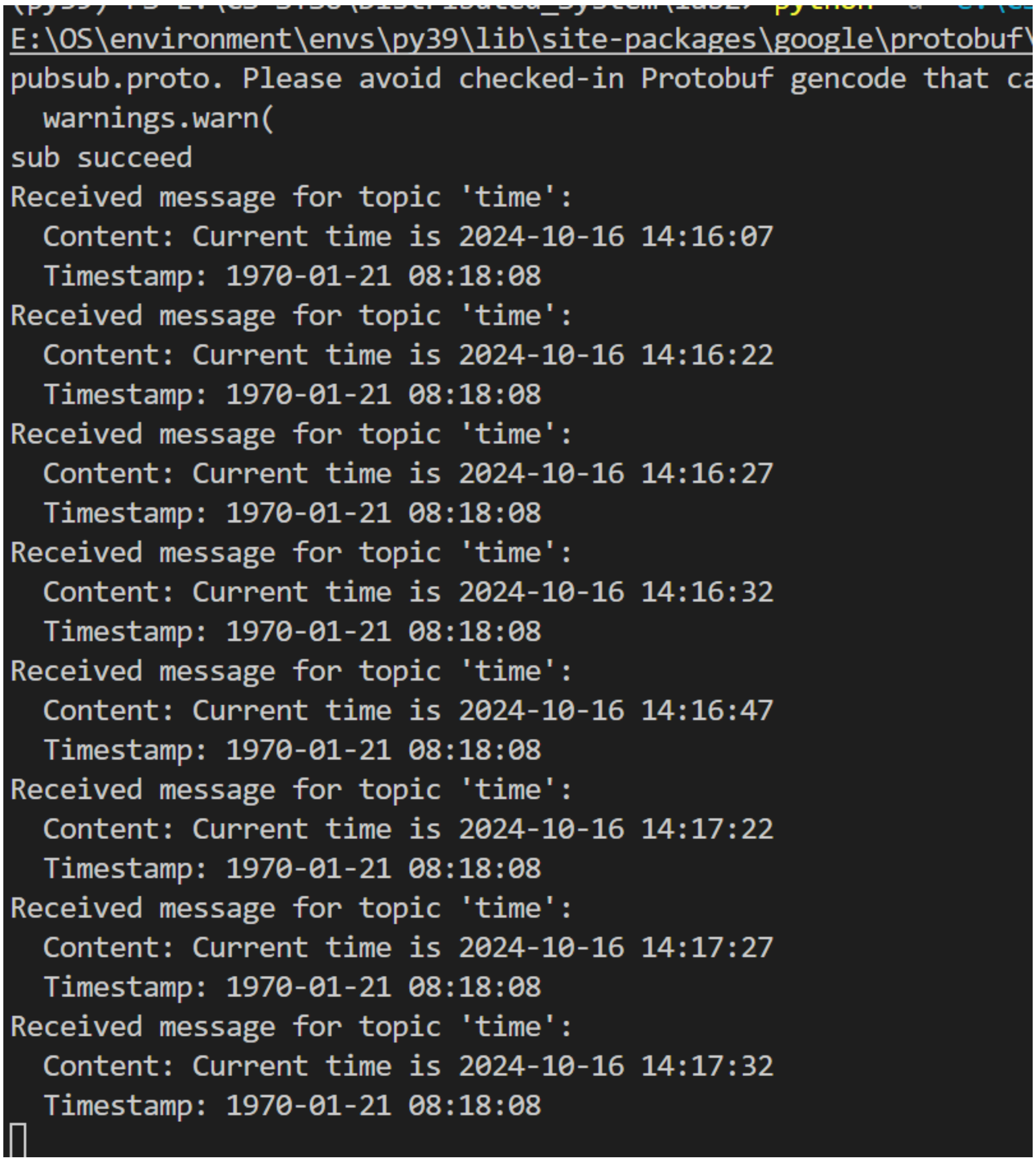
```
(py39) PS E:\CS-SYSU\Distributed_System\lab2> python -u "e:\CS-SYSU\Distr
E:\OS\environment\envs\py39\lib\site-packages\google\protobuf\runtime_ver
pubsub.proto. Please avoid checked-in Protobuf gencode that can be obsole
  warnings.warn(
sub succeed
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:12:43
  Timestamp: 1970-01-21 08:18:07
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:12:48
  Timestamp: 1970-01-21 08:18:07
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:12:53
  Timestamp: 1970-01-21 08:18:07
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:12:58
  Timestamp: 1970-01-21 08:18:07
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:13:04
  Timestamp: 1970-01-21 08:18:07
```

- 检验两个客户端分别订阅两个不同的话题：
  服务器：自动生成两个话题消息



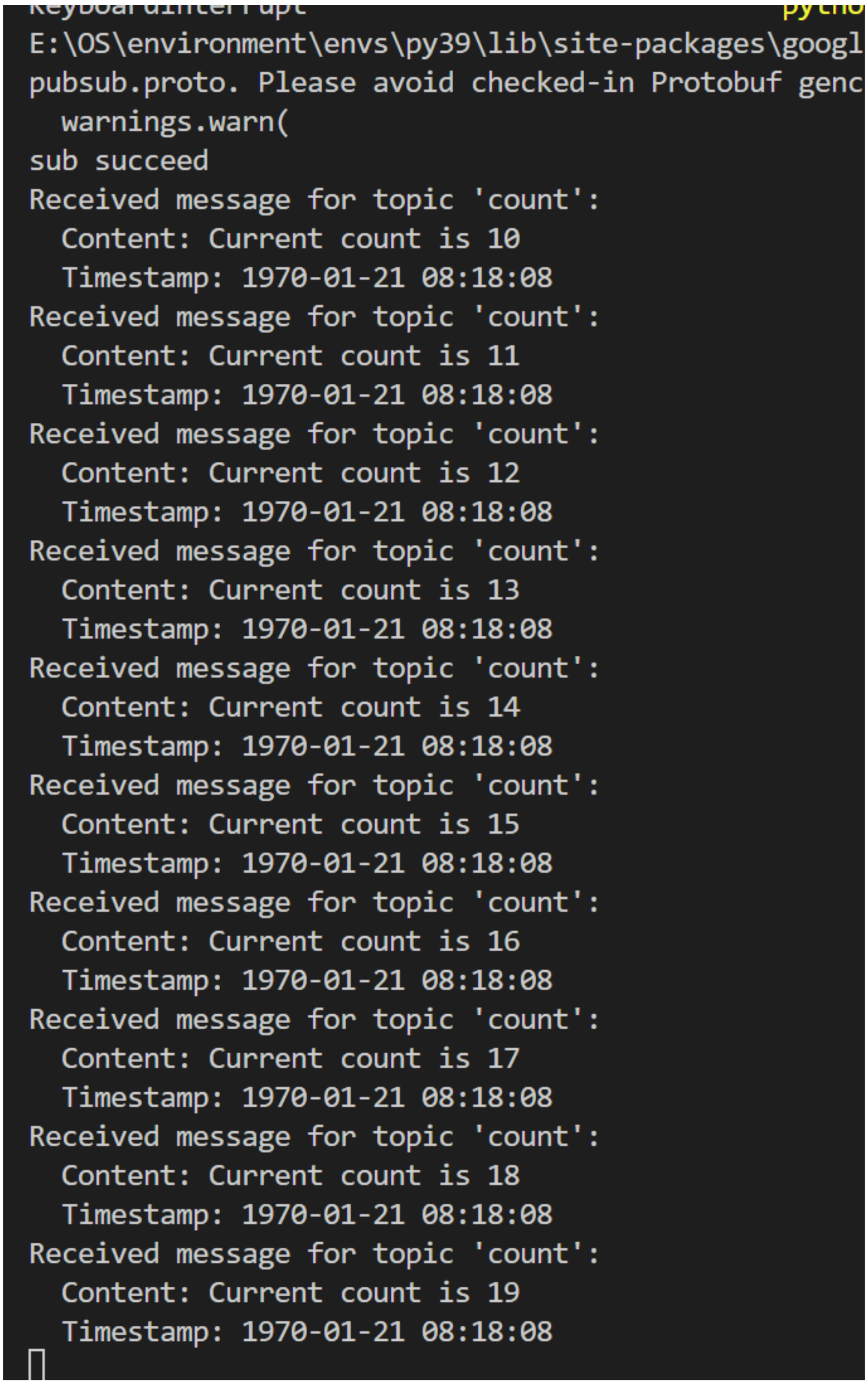
```
(py39) PS E:\CS-SYSU\Distributed_System\lab2> python -u "e:\CS-SYSU\Distribute
E:\OS\environment\envs\py39\lib\site-packages\google\protobuf\runtime_version
pubsub.proto. Please avoid checked-in Protobuf gencode that can be obsolete.
  warnings.warn(
Server started, listening on 40000
add topic time: Current time is 2024-10-16 14:16:07 1729088167
add topic count Current count is 10 1729088172
add topic count Current count is 11 1729088177
add topic time: Current time is 2024-10-16 14:16:22 1729088182
add topic time: Current time is 2024-10-16 14:16:27 1729088187
add topic time: Current time is 2024-10-16 14:16:32 1729088192
add topic count Current count is 12 1729088197
add topic count Current count is 13 1729088202
add topic time: Current time is 2024-10-16 14:16:47 1729088207
add topic count Current count is 14 1729088212
add topic count Current count is 15 1729088217
add topic count Current count is 16 1729088222
add topic count Current count is 17 1729088227
add topic count Current count is 18 1729088232
add topic count Current count is 19 1729088237
add topic time: Current time is 2024-10-16 14:17:22 1729088242
add topic time: Current time is 2024-10-16 14:17:27 1729088247
add topic time: Current time is 2024-10-16 14:17:32 1729088252
```

  客户端1：订阅time话题消息并且得到time话题中更新的消息



```
[illegible]
E:\OS\environment\envs\py39\lib\site-packages\google\protobuf\
pubsub.proto. Please avoid checked-in Protobuf gencode that ca
  warnings.warn(
sub succeed
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:16:07
  Timestamp: 1970-01-21 08:18:08
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:16:22
  Timestamp: 1970-01-21 08:18:08
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:16:27
  Timestamp: 1970-01-21 08:18:08
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:16:32
  Timestamp: 1970-01-21 08:18:08
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:16:47
  Timestamp: 1970-01-21 08:18:08
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:17:22
  Timestamp: 1970-01-21 08:18:08
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:17:27
  Timestamp: 1970-01-21 08:18:08
Received message for topic 'time':
  Content: Current time is 2024-10-16 14:17:32
  Timestamp: 1970-01-21 08:18:08
```

客户端2：订阅count话题，并且得到发送请求以来所有的更新消息



```
KeyboardInterrupt                                                      pytho
E:\OS\environment\envs\py39\lib\site-packages\googl
pubsub.proto. Please avoid checked-in Protobuf genc
  warnings.warn(
sub succeed
Received message for topic 'count':
  Content: Current count is 10
  Timestamp: 1970-01-21 08:18:08
Received message for topic 'count':
  Content: Current count is 11
  Timestamp: 1970-01-21 08:18:08
Received message for topic 'count':
  Content: Current count is 12
  Timestamp: 1970-01-21 08:18:08
Received message for topic 'count':
  Content: Current count is 13
  Timestamp: 1970-01-21 08:18:08
Received message for topic 'count':
  Content: Current count is 14
  Timestamp: 1970-01-21 08:18:08
Received message for topic 'count':
  Content: Current count is 15
  Timestamp: 1970-01-21 08:18:08
Received message for topic 'count':
  Content: Current count is 16
  Timestamp: 1970-01-21 08:18:08
Received message for topic 'count':
  Content: Current count is 17
  Timestamp: 1970-01-21 08:18:08
Received message for topic 'count':
  Content: Current count is 18
  Timestamp: 1970-01-21 08:18:08
Received message for topic 'count':
  Content: Current count is 19
  Timestamp: 1970-01-21 08:18:08
```