CSCI 5105 - Project 1 PubSub

Tiannan Zhou <zhou0745@umn.edu>, Xuan Bi <bixxx078@umn.edu>

Feb 14, 2018

1. Project Description

There are four main files generated by us. They are *client.cpp groupServer.cpp pubsub.h* and a makefile to compile and run the project program.

1.1 makefile

This is the file to compile and generate the group server and client or them separately. It can also clean the two servers and some test purposed script.

1.2 pubsub.h

This file includes all the necessary headfiles, functions to check whether there is a valid request to unsubscribe/subscribe an article or to publish an article by checking the input article format. It also includes the helper functions to get the local IP address and simply sending packet via UDP.

1.3 groupServer.cpp

This is the file to realize servers. It has multiple implementations. GroupServer needs to receive from registry server the heartbeat, we will use this thread to receive response from Registry Server for GetList() as well. Additionally, it needs functionalities of getting article and client indexes in each request. For each request, server would check if the client has subscribed the article. If not, article should not be sent to the client; otherwise, it has an implementation to publish it. In the meantime, client needs to join a server or leave and finally unsubscribe/subscribe articles or publish one. During each of the operations, one needs to lock up the clients related data to prohibit the deadlock situation even though it is unlikely in this case since it is thread safe herein.

1.4 client.cpp

This is the file to realize the functionalities of client to leave/exit the server, unsubscribe/subscribe/publish articles, and to ping whether the server is up by RPC. And the functionality to receive the UDP packet as article's publication is included here as well.

2. Functionality for Components

2.1 GroupServer

2.1.1 Listening to RPC calls

We've set up six functions to support RPC call. They're join, leave, subscribe, unsubscribe, publish and ping. For joining, the server will register this client whether this is the first-time connection or re-connecting. For leave, the server will deregister this client and clean all subscriptions related to this client. For subscribe, the server will add this subscription to the client's subscription list and check duplication at the same time to keep subscription list organized. For unsubscribe, the server will check whether this client subscribed the same category before, if yes, the server will remove this subscription. For publish, the server will deliver it to all clients who have subscribed match category. For ping, the server will return executed successfully directly (we use (int) 1 to represent executed successfully). For every call, we used the return value as the execution status to let the client know what happened on the server side. Here's the meaning for RPC remote function's return value:

0: unknown error

1: executed successfully

2: duplicate connection request

3: connection has not established

4: duplicate subscribe request for a client

5: reached the MAXSUBSCRIPE for certain client

6: reached the limit for connected client

7: unsubscribe a non-existent subscription

8: invalid subscribe or publish request

2.1.2 Thread to Handle Server-side Operations

At the beginning of server running, the server will create a thread to listen to the command line's input. We support two operations: the first one is GetList(), which is sending request to Registry Server for getting the list of connected group servers. But please note, the return value won't be received in this thread, the response from

Registry will be received the thread for Receiving Messages from Registry Server (descripted in 2.1.3) and that thread will pass the response received to here via usage of shared memory and conditional variable. The second one is Deregister(). By the way, Register() will happen automatically while the server starts. The group server will only call Register() at the beginning of this server and call Deregister() before exiting.

2.1.3 Receiving Messages from Registry Server

The server will create a thread to bind a specific port to listen to every message sent from Registry Server. If the message is “heartbeat”, this thread will reply “heartbeat” directly to the Register Server’s port 5105. And if received any messages other than “heartbeat ”, it would assume this is response for GetList() request and pass it to shared memory, then sending a thread signal to the conditional variable let GetList know the response has been back.

2.1.4 Signal Binding

The server will bind signal SIGINT to a function that sending Deregister() to Registry Server and exit. This could let Registry Server know this server is going to be down although Registry Server would eventually know it by heartbeat.

2.2 Client

2.2.1 Sending Request to Call Remote Functions

The client’s main thread will use command line to listen to user’s input and call 6 different remote functions. The client will do local validness check before calling functions.

2.2.2 Receiving Articles from GroupServer

The client will create a thread the bind a port to listen to any incoming transmissions after user specified the port which to receive articles. This thread will keep listening any incoming messages and printing it out to the screen.

3. How to Compile and Run the Project

3.1 Making

We wrote a makefile to simplify the compilation and execution processes. To successfully compile and run the code, simply run the following command in the directory where the *makefile* is located.

$ make

After making, you will see two expectative file named “GroupServer” and “Client” in your directory

3.2 Run GroupServer

$ ./GroupServer

to start the server. Remember you must enter the port number you want to use communicate with the Register Server as the instruction, then you can start to use server’s functionality. There are two operations that are supported in the command line: “1. GetList()” and “2.Deregister() and exit the server”. You can enter the corresponding operation number to call functions. And log information for functions that remotely called by clients will be printed on the screen as well.

3.3 Run Client

$ ./Client

to start the client. The client supports 6 different remote function call to Group Server and an exit option with cleaning everything well. Firstly, you need to specify a port to receive articles then entering the Group Server’s IP address. Remember you must join a launched Group Server to let the functionality work properly. You will see an instruction after the connection to Group Server has been established and please enter the operation number to use different functions. NOTE: for subscribe, unsubscribe and publish, you need to enter standardized article as the instruction descripted to call the corresponding function. The client supports entering spaces within the content (which means you can send contents with spaces) and the article should be finished with hitting Return. All running result for each call will be displayed on the screen. If you want to exit, enter 9 in the main menu.

4. Testing Cases and Results

4.1 single client and server

This is the simplest case where there is no interaction between the clients.

Opening a group server and a client on an IP. First to ping to test if server is up.

You have connected to the server 128.101.37.27

0. Ping

1. Leave

2. Subscribe

3. Unsubscribe

4. Publish

9. Exit

Please enter the operation #: 0

Pinged successfully

Then to try subscription:

Enter the article you want to subscribe: Science;;;

Your subscribe request <Science;;;> has been sent successfully

Now assume it publishes an article in Science:

Enter the article you want to publish: Science;zhou;MIT;CS

!!!ARTICLE!!!: received article <Science;zhou;MIT;CS> from 128.101.37.27:50505

start receiving articles by Port [8888]

Your publish request <Science;zhou;MIT;CS> has been sent successfully

The published article is automatically sent to the client since it subscribed.

This client has successfully received this article. Now we try to unsubscribe it in two ways:

Enter the article you want to unsubscribe: Science;;;

Your unsubscribe request <Science;;;> has been sent successfully

Enter the article you want to unsubscribe: ;Tiannan Zhou;;

ERROR: unsubscribe a non-existent subscripition

The server cannot process your request, try again later.

It shows that if one unsubscribes it by the way one subscribed, it works though successfully but not so if one does it in different ways from the original subscription.

The client now can no longer receive any articles.

4.2 multiple clients and single server

Now to see how different client interact with one another. Two clients first subscribe their articles individually then try to publish articles.

Enter the article you want to publish: Health;;UPenn;nutrition

Your publish request <Health;;UPenn;nutrition> has been sent successfully

!!!ARTICLE!!!: received article <Health;;UPenn;nutrition> from 128.101.37.27:53087

start receiving articles by Port [8888]

The server shows this transaction as follows:

Client 128.101.35.147:8888 connected to the server

Client 128.101.35.147:8888 subscribed <Health;;UPenn;>.

Client 128.101.35.49:8888 published <;;UPenn;nutrition>

Client 128.101.35.49:8888 published <Health;;UPenn;nutrition>

Sent <Health;;UPenn;nutrition> to 128.101.35.147:8888

We see that the published article should contain exact keywords the other client subscribes in order to be sent to it.

Furthermore, once a server leaves a server, cache for its information will be cleaned up.

Client 128.101.35.49:8888 left the server, clean all cache for it

4.3 negative testcases

In this section, we will use some more negative cases to show the GroupServer and Client's error handling ability.

4.3.1 Trying to send an invalid subscription request:

Please enter the operation #: 2

Enter the article you want to subscribe: ;jon;;somecontents

ERROR: Your input article subscribe request is invalid

The client figured out it's an invalid request and prohibited its sending.

4.3.2 Sending a duplicated subscription request:

On Client side:

Please enter the operation #: 2

Enter the article you want to subscribe: ;;UMN;

Your subscribe request <;;UMN;> has been sent successfully

Please enter the operation #: 2

Enter the article you want to subscribe: ;;UMN;

ERROR: duplicate subscribe request for a client

The server cannot process your request, try again later.

On GroupServer side:

Client 128.101.38.193:7749 subscribed <;;UMN;>.

Client 128.101.38.193:7749 tried to subscribe <;;UMN;>, but it's a duplicate subscribe.

It's easy to see that GroupServer found the subscription request is a duplicate one and didn't insert it into the subscription list. It also let the client know it is duplicate via the return value for remote function call.

4.3.3 Unsubscribing a non-existent subscription:

This has already been shown in section 4.1.

4.3.4 Publishing an invalid article:

Please enter the operation #: 4

Enter the article you want to publish: ;;;notitlepaper

ERROR: Your article publish request is invalid

It looks that this article without any information's publishing request has been suspended by the client side.

5. Comparison with Google PubSub

With well-implemented API from Google PubSub. Regardless of subscribing, unsubscribing or publishing, all operations are becoming much easier to be developed from developers' perspective. And developers could use these APIs to develop applications with different functionality requirement easily. But for our original implementation, it's hard to make any

change if the development requirement was modified. Good well-designed and implemented APIs are saving tons of time in development.