A full description of all functions code used can be found in the html generated from doxygen. Open index.html within the html folder in a browser to view the html describing the project. (I didn't include the images or html in page length… hope that's OK)

Client Side Code

- Client has a thread to listen for incoming UDP publications from the server.
- Received articles are saved to the disk.
- Client sends a UDP verification to the server after receiving an article.
- Client updates upon launching or after a server connection is re-established.
- Server IP, server port, and client's listening port are permanently set in the code.
- Command-line entering is possible by uncommenting specific lines of code.
- Client confirms server-port combination with an initial ping and continues trying to access the server if it fails.
- Client uses the first non-loopback IP for automatic update but can manually update with any valid IP and port.
- Mutual exclusion locks are in place to prevent race conditions.
- Publications should not arrive out of order due to verifications being waited for on the server side before any additional publications can be sent by the server.
- Debugging information about failed RPC calls is logged on the server-side.
- If the server connection fails, the client locks down manual commands, attempts to reconnect every 3 seconds, and updates upon successful reconnection.
- Unlocking manual calls outside the ping thread after connection re-establishment may cause client crashes, so manual calls stay locked until the application restarts. The client update and restarting of the listening thread for new publications is still operational.

Server Side Code

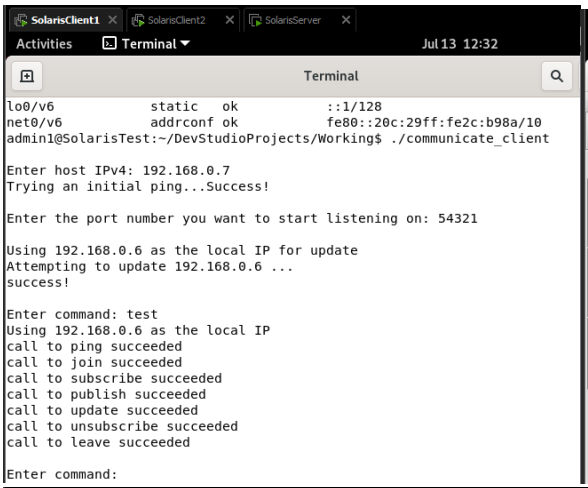
- Client, subscription, and failed send information is kept in memory and saved to disk upon any changes. Disk is read from during server initialization.
- No strict limit on the number of subscriptions or unverified sent messages.
- Mutual exclusion locks are in place to prevent race conditions.
- Full logging of all calls, failures, and debugging information is kept in server.log.
- Logging of verifications received and timeouts is saved in verification.log.
- Joined clients are stored in a char array.
- Subscriptions are stored in pubSub structures with an address structure.

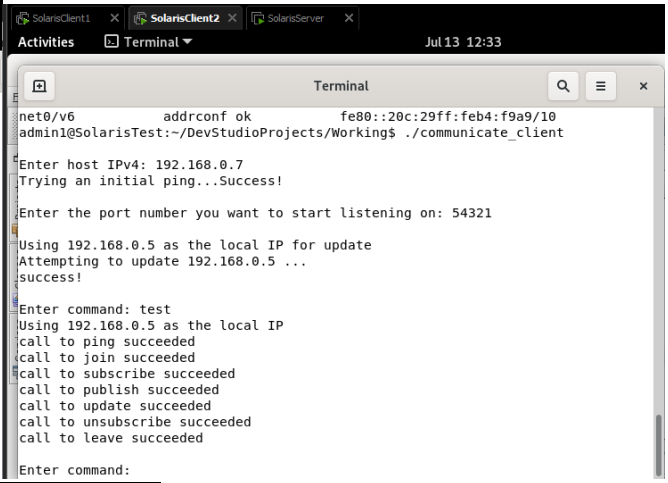
- Hash values are made for both the address and full object, but they have not been implemented into searches.
- Articles must match publication and subscription requirements per assignment instructions, but the application can handle incorrect input well.
- Publications are sent only to clients that meet the subscription data and are actively joined.
- Blank space in articles is treated as a wildcard.
- Only one article is sent to each client even if they have multiple matching subscriptions.
- More general subscriptions supersede more specific subscriptions in determining if an article is sent.

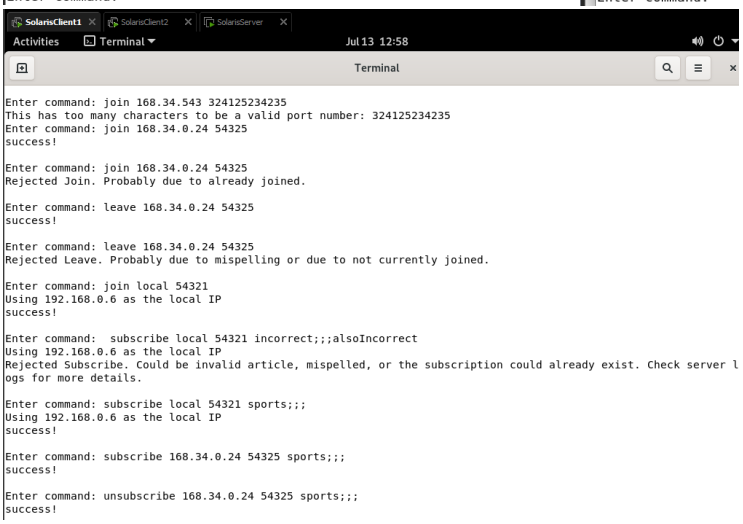
Demonstration

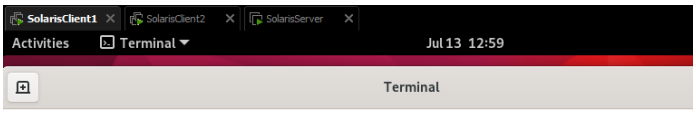
The images on the following pages provide illustration for this demonstration.

1. Start the server.
2. Startup Client 1 with IP 192.168.0.6 and Client 2 with IP 192.168.0.5.
3. Call the test program on both clients.
4. Client 1 demonstrates resilience to different and incorrect RPC calls.
5. Stop and restart the server.
6. Client 1 detects server unresponsiveness, locks down manual commands, and checks for server responsiveness every 3 seconds.
7. Restart Client 2, join, subscribe, and publish an article.
8. Client 1 receives the article despite server restart while client application was active.
9. Shutdown Client 1 and have Client 2 publish an article intended for both clients.
10. Client 1 doesn't send a verification due to being inactive. Server saves failed send information. Upon restart, Client 1 automatically updates.

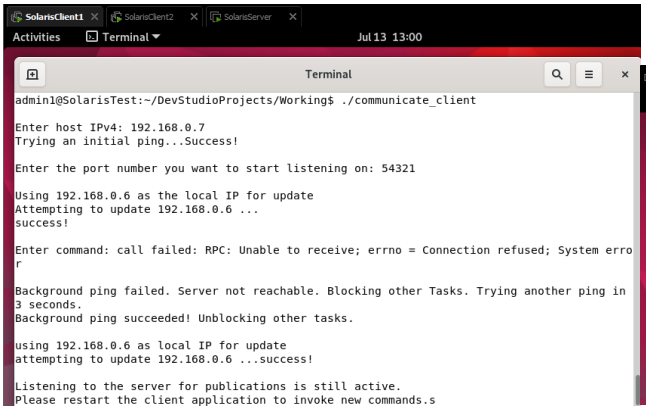


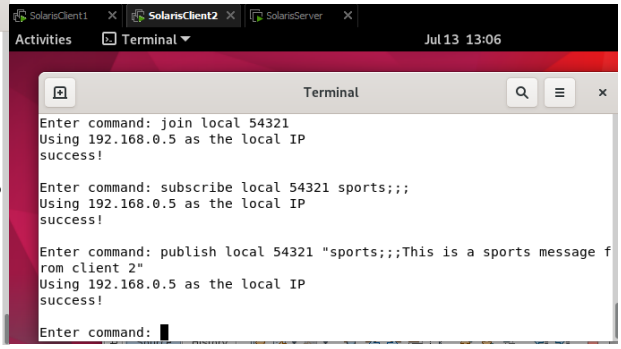


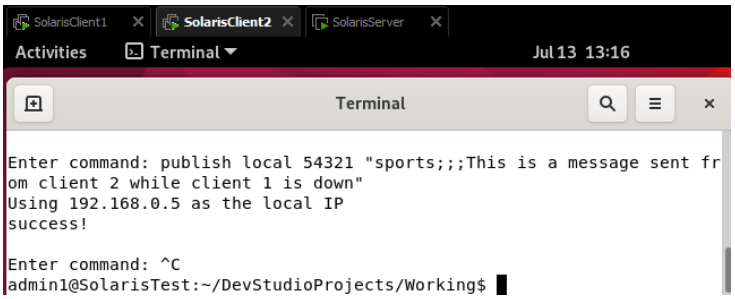


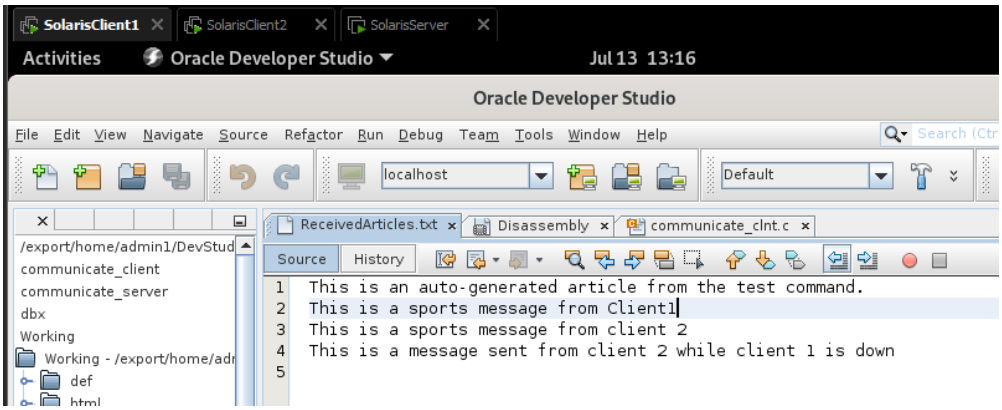


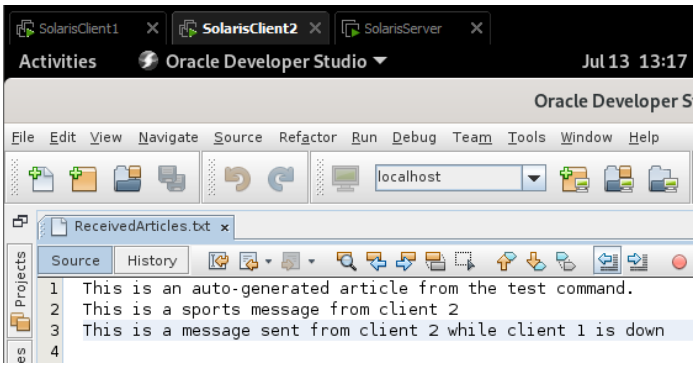

```
Enter command: publish local 54321 "sports;;;This is a sports message from Client1"
Using 192.168.0.6 as the local IP
success!
```

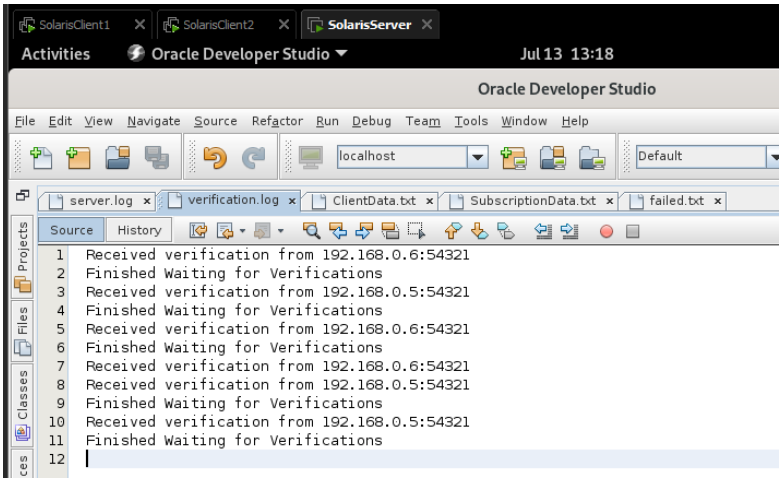


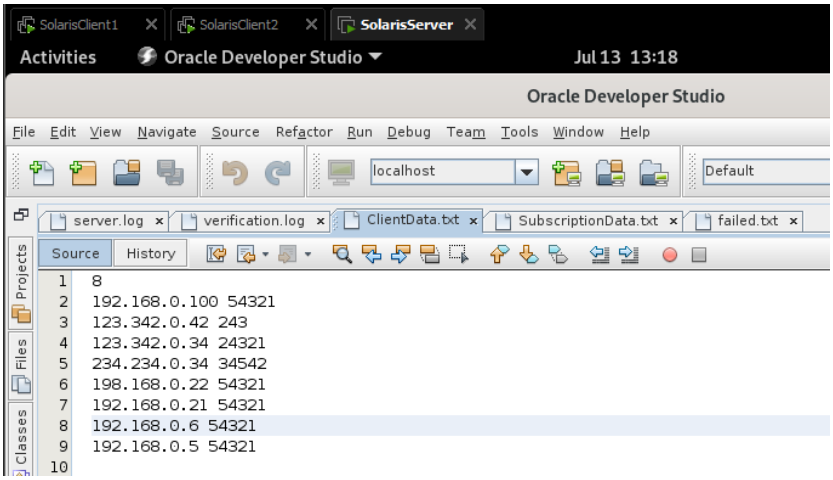


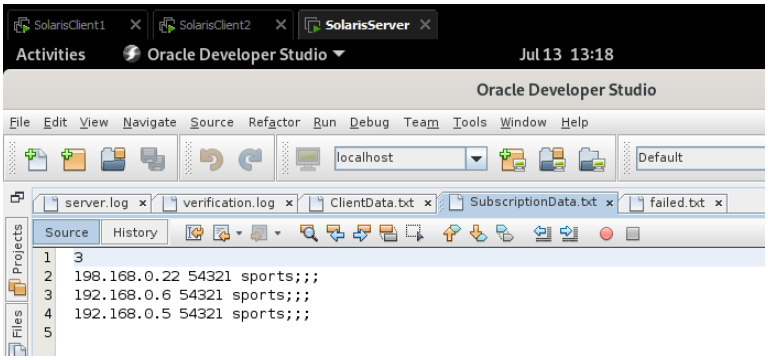

SolarisClient1
SolarisClient2
SolarisServer
Activities
Terminal
Jul 13 13:16
Terminal
Enter command: publish local 54321 "sports;;;This is a message sent fr
om client 2 while client 1 is down"
Using 192.168.0.5 as the local IP
success!
Enter command: ^C
admin1@SolarisTest:~/DevStudioProjects/Working$

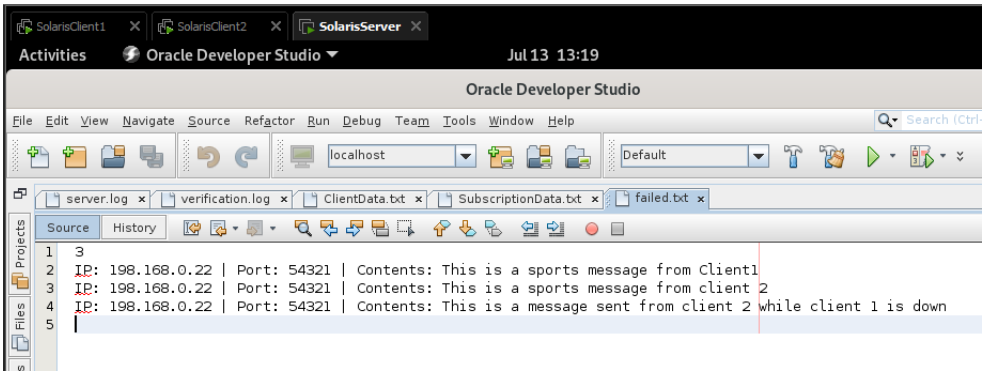

SolarisClient1
SolarisClient2
SolarisServer
Activities
Oracle Developer Studio
Jul 13 13:16
Oracle Developer Studio
File Edit View Navigate Source Refactor Run Debug Team Tools Window Help
localhost
Default
ReceivedArticles.txt
Disassembly
communicate_clnt.c
Source
History
/export/home/admin1/DevStud
communicate_client
communicate_server
dbx
Working
Working - /export/home/adr
def
1 This is an auto-generated article from the test command.
2 This is a sports message from Client1
3 This is a sports message from client 2
4 This is a message sent from client 2 while client 1 is down
5

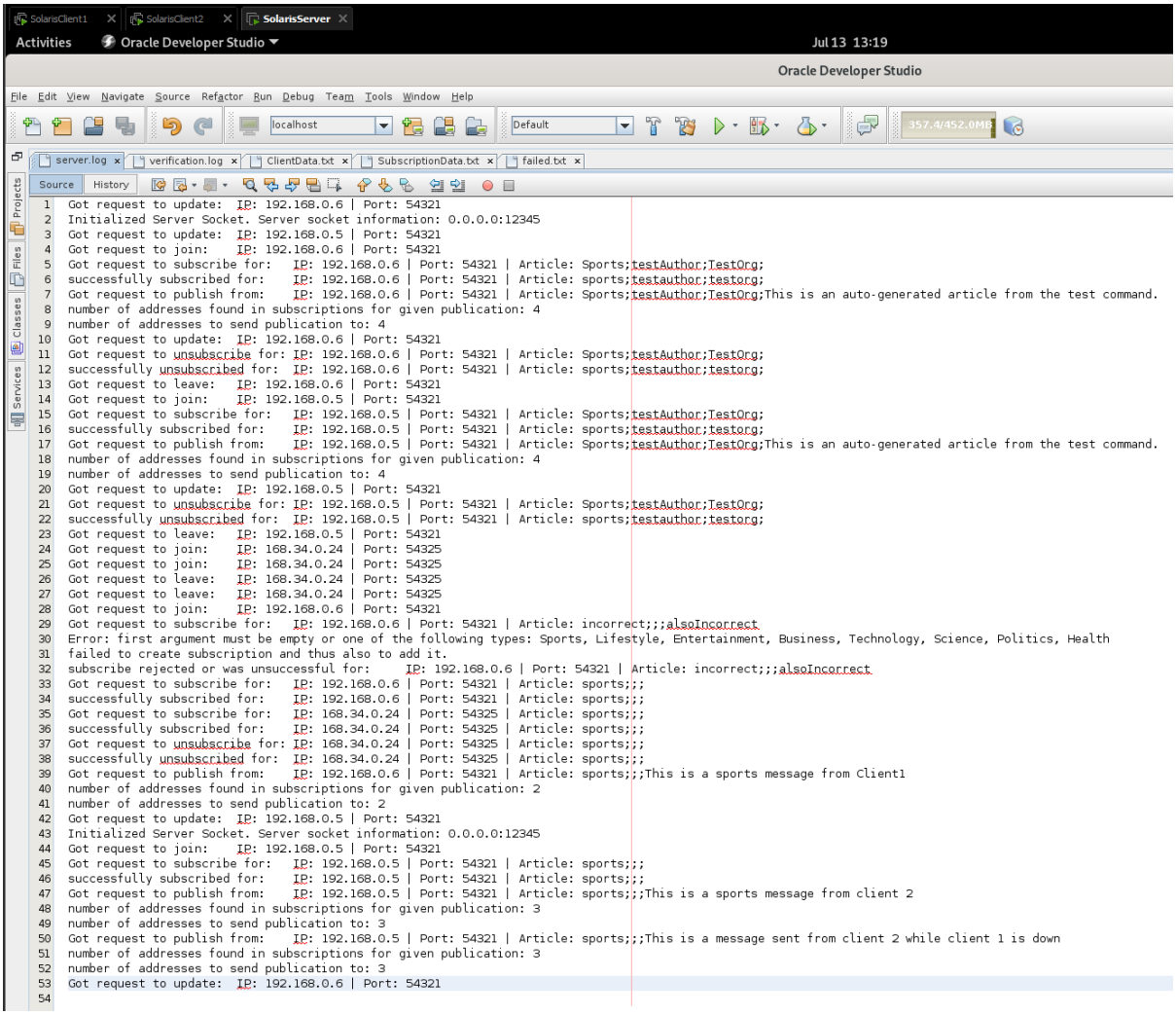

SolarisClient1
SolarisClient2
SolarisServer
Activities
Oracle Developer Studio
Jul 13 13:17
Oracle Developer S
File Edit View Navigate Source Refactor Run Debug Team Tools Window Help
localhost
ReceivedArticles.txt
Source
History
Projects
1 This is an auto-generated article from the test command.
2 This is a sports message from client 2
3 This is a message sent from client 2 while client 1 is down
4


SolarisClient1
SolarisClient2
SolarisServer
Activities
Oracle Developer Studio
Jul 13 13:18
Oracle Developer Studio
File Edit View Navigate Source Refactor Run Debug Team Tools Window Help
localhost
Default
server.log
verification.log
ClientData.txt
SubscriptionData.txt
failed.txt
Source
History
Projects
Files
Classes
1 Received verification from 192.168.0.6:54321
2 Finished Waiting for Verifications
3 Received verification from 192.168.0.5:54321
4 Finished Waiting for Verifications
5 Received verification from 192.168.0.6:54321
6 Finished Waiting for Verifications
7 Received verification from 192.168.0.6:54321
8 Received verification from 192.168.0.5:54321
9 Finished Waiting for Verifications
10 Received verification from 192.168.0.5:54321
11 Finished Waiting for Verifications
12


SolarisClient1
SolarisClient2
SolarisServer
Activities
Oracle Developer Studio
Jul 13 13:18
Oracle Developer Studio
File Edit View Navigate Source Refactor Run Debug Team Tools Window Help
localhost
Default
server.log
verification.log
ClientData.txt
SubscriptionData.txt
failed.txt
Source
History
Projects
Files
Classes
1 8
2 192.168.0.100 54321
3 123.342.0.42 243
4 123.342.0.34 24321
5 234.234.0.34 34542
6 198.168.0.22 54321
7 192.168.0.21 54321
8 192.168.0.6 54321
9 192.168.0.5 54321
10

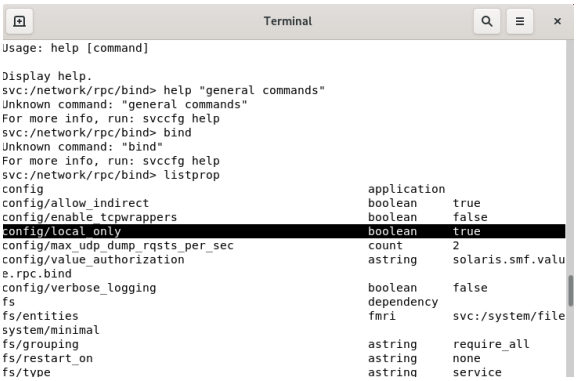

SolarisClient1
SolarisClient2
SolarisServer
Activities
Oracle Developer Studio
Jul 13 13:18
Oracle Developer Studio
File Edit View Navigate Source Refactor Run Debug Team Tools Window Help
localhost
Default
server.log
verification.log
ClientData.txt
SubscriptionData.txt
failed.txt
Source
History
Projects
Files
1 3
2 198.168.0.22 54321 sports;;;
3 192.168.0.6 54321 sports;;;
4 192.168.0.5 54321 sports;;;
5


SolarisClient1
SolarisClient2
SolarisServer
Activities
Oracle Developer Studio
Jul 13 13:19
Oracle Developer Studio
File Edit View Navigate Source Refactor Run Debug Team Tools Window Help
localhost
Default
server.log
verification.log
ClientData.txt
SubscriptionData.txt
failed.txt
Source
History
Projects
Files
1 3
2 IP: 198.168.0.22 | Port: 54321 | Contents: This is a sports message from Client1
3 IP: 198.168.0.22 | Port: 54321 | Contents: This is a sports message from client 2
4 IP: 198.168.0.22 | Port: 54321 | Contents: This is a message sent from client 2 while client 1 is down
5



Final Note:

I had to try a ton of different things to get rpcgen to work properly and compile after generation. I even used a bunch of different operating systems. I eventually found Solaris worked to compile, but I had to do a bunch of debugging as to why communication across different VMs wasn't possible. I found the following very obscure setting needed to be manually changed after sifting through very granular log files. I wouldn't recommend using rpcgen anymore to anyone for rpc programming.



This can be changed with:
sudo svccfg
select network/rpc/bind
setprop -G config -P local_only -T "boolean" "false"
listprop