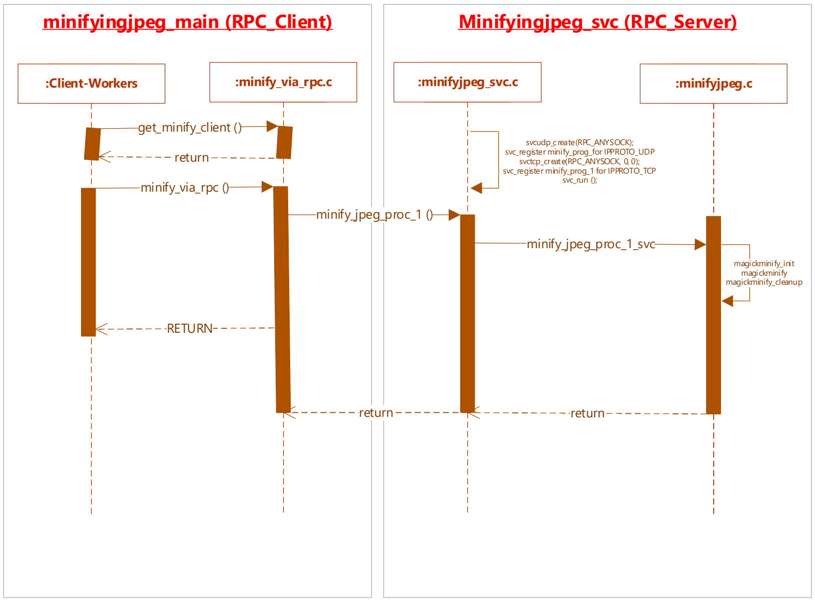
Part 1

Part 1 of Project #4 implementation was straight forward of understanding the flow of the request from Client to Server and response back to client.

Basic steps for creating project are:

1. Define XDR datatypes needed for the RPC to communicate between Server and Clients.
2. Define the RPC Method for the client to use to communicate with RPC Server.
3. Run RPCGen
4. Compile
5. Update the client and server to call magikminify library calls.



Part 2

Initial design

Summary:

Part 2 Since it uses multi-threaded decided to use the same known pattern and existing code base from P1 to P3.

Boss Worker thread pattern with each service to communicate.

1 Message queues for workers to communicate with boss on a RPC given request.

```
main (int argc, char **argv)
```

- Added a thread count options to create ‘t’ worker threads
- minify_prog_1() with svc_register() for RPC request handler.
- Creation and Free of threads and STEQUE resources.
- svc_run ()

```
static void minify_prog_1(struct svc_req *rqstp, register SVCXPRT *transp)
```

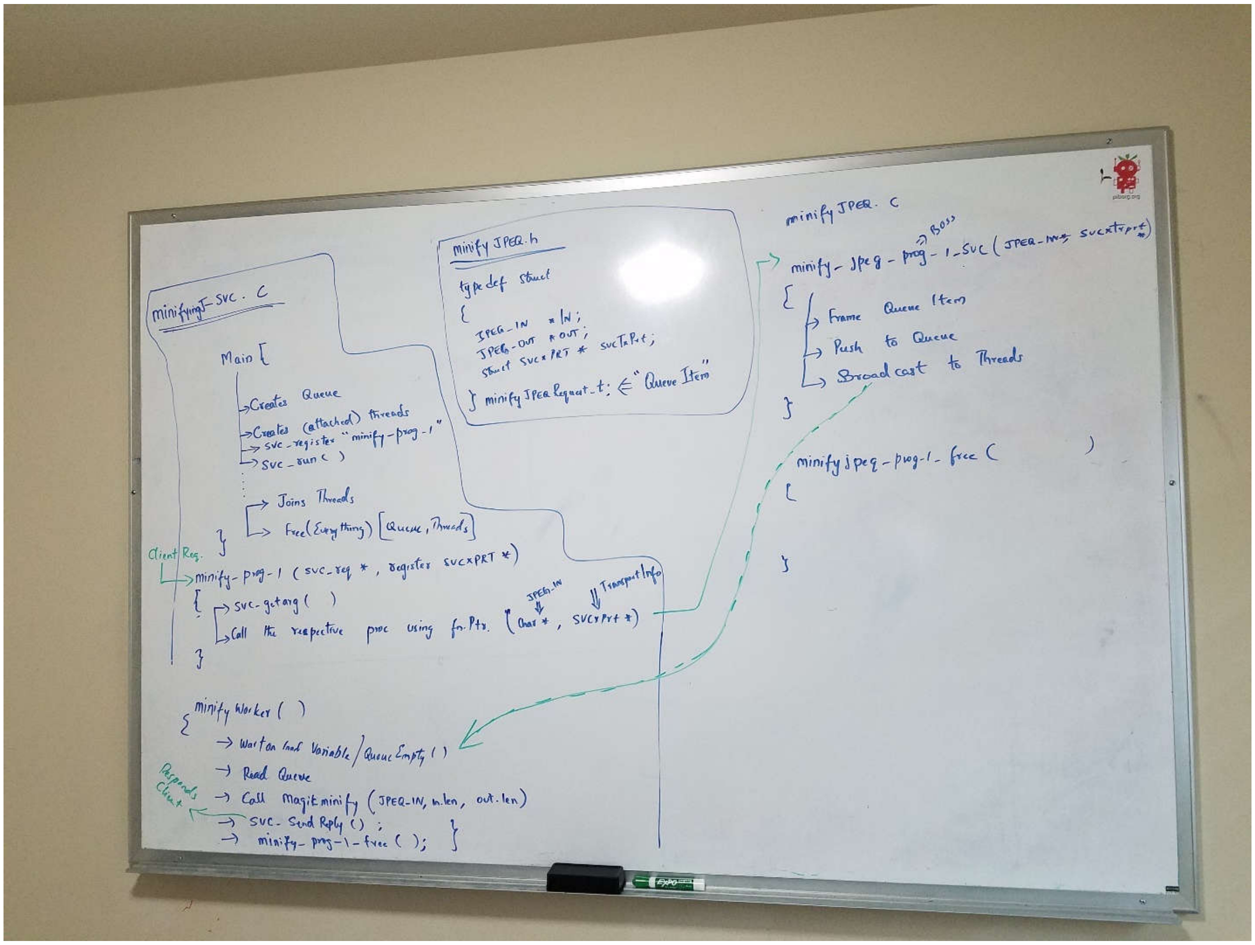
- Initialize a function pointer and calls without parsing svc args ()

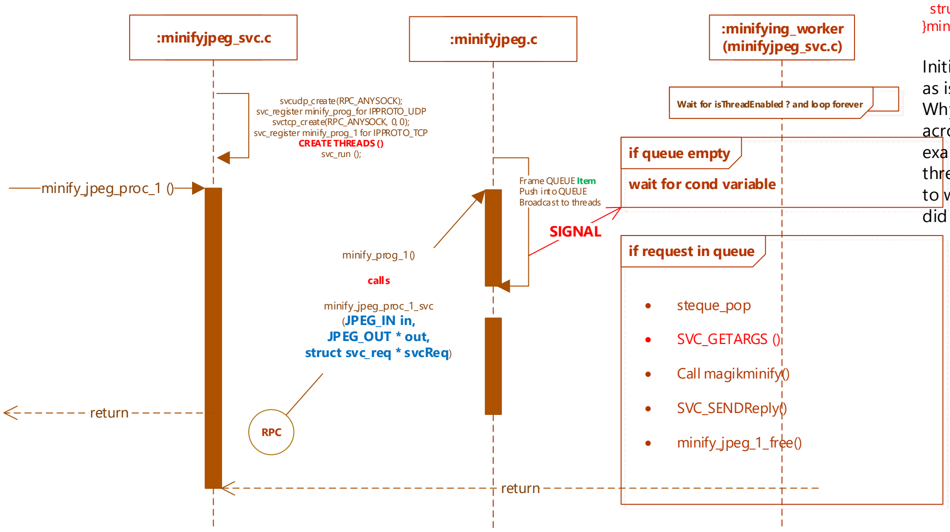
```
bool_t minify_jpeg_proc_1_svc(JPEG_IN in, xdrproc_t *xdrArg, struct svc_req
*svcReq, SVCXPRT *transp)
```

- Push the svc_arg without parsing and push in to a queue and broadcast signals to threads

```
void *MinifyWorker(void* arg)
```

- Reads from the queue
- svc_getargs()
- Parse and invoke magikminify () to minify
- svc_sendreply ()
- minify_prog_1_freeresult ()


minifyingJ-svc . c
Main {
→ Creates Queue
→ Creates (attached) threads
→ svc_register "minify-prog-1"
→ svc_run ( )
→ Joins Threads
→ Free(Everything) [Queue, Threads]
}
Client Req.
→ minify-prog-1 (svc_req * , register SVCXPRT *)
{ → svc-getarg ( )
→ Call the respective proc using fn. Ptr. (Char * , SVCxPrt *)
JPEG_IN
Transport Info
}
minify JPEQ.h
typedef struct
{
JPEG_IN *IN;
JPEG-OUT *OUT;
struct SVCxPRT * svcTxPrt;
} minify JPEQ Request_t; ⇐ "Queue Item"
minify JPEQ. C
minify- Jpeg - prog - 1_svc (JPEQ-IN, svcxtrprt *)
Boss
{
→ Frame Queue Item
→ Push to Queue
→ Broadcast to Threads
}
minify jpeg-prog-1_free ( )
{
}
minify Worker ( )
{
→ Wait on Cond Variable / Queue Empty ( )
→ Read Queue
→ Call magik minify (JPEQ-IN, in.len, out.len)
→ svc- Send Reply ( ) ;
→ minify- prog-1- free ( ); }
Responds Client



```
typedef struct {
  SVCXPRT        *transp;
  struct svc_req  *svcReq;
}minifyRequest_t;
```

Initially I used RPC request parameters as is to pass all the way to thread. Why, I read the readme, but still came acrossa a redhat archived article with example on calling svc_getargs() in thread for each request, I felt its easy to write and finish than Part#2 design I did later.

Why:

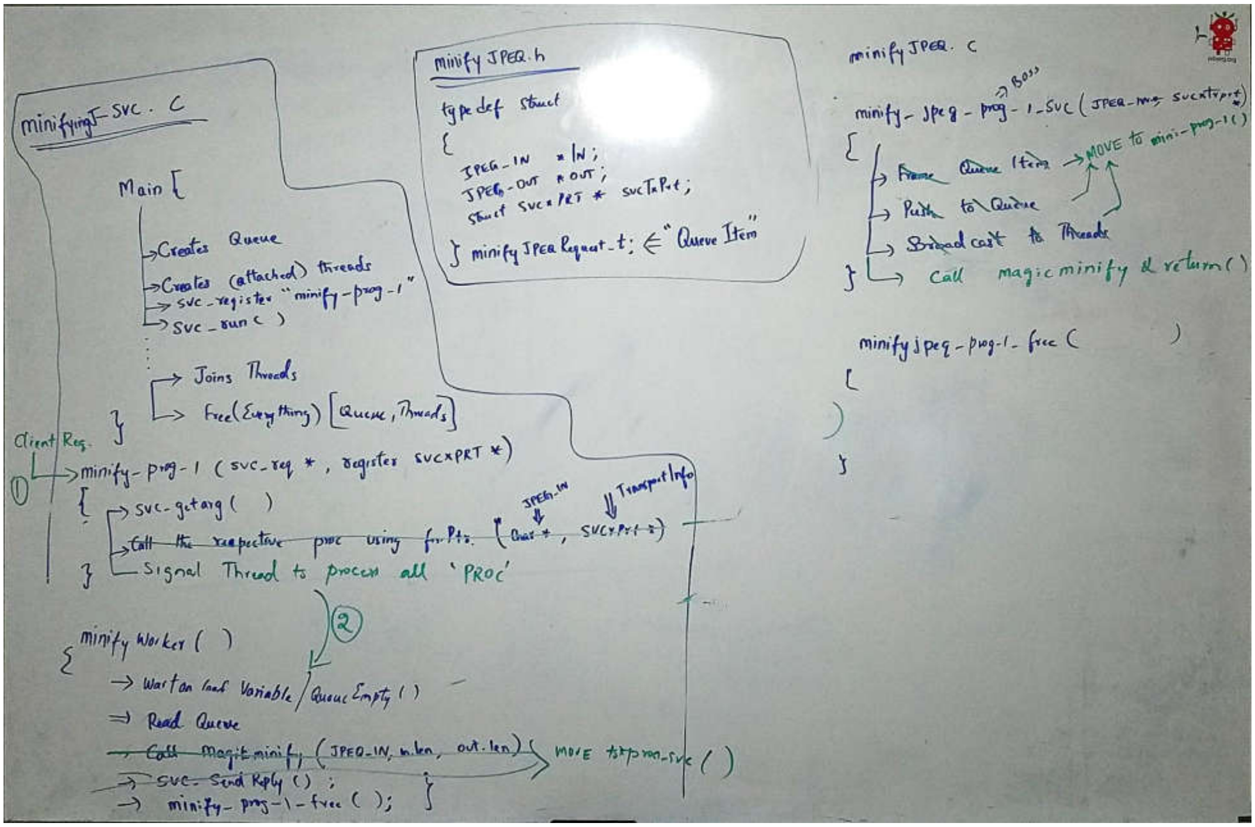
Passing the svc request arguments all the way to the workers make it clean and simple in terms of implementation. Also, it allows the RPC_Server to accept another request right away, improved perf.

Issues:

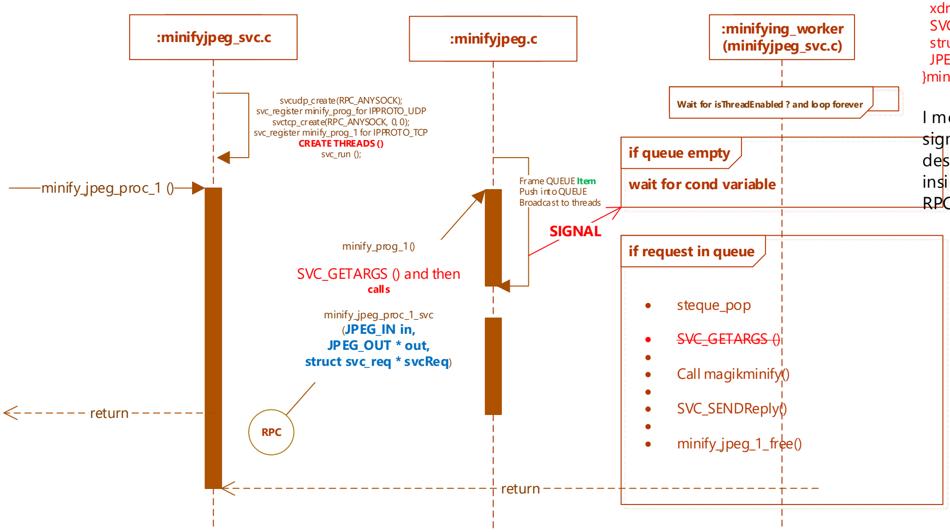
But I got RPC: Server can't decode arguments" and RPC: Server can't encode arguments" for UDP and TCP. I followed https://www.redhat.com/archives/redhat-list/2004-June/004164.html this as a reference and overriding what it says in project#4 readme of svc_getargs().

Second design

So, I moved the svc_getargs () in to static void minify_prog_1( ) and modified my queue item members to accommodate new arguments needed for the worker to post RPC response.



```
minifyingJ-svc . c

    Main {
      ->Creates  Queue
      ->Creates  (attached) threads
      -> svc_register "minify-prog-1"
      -> svc_run ( )
      :
      -> Joins Threads
      -> Free(Everything) [Queue, Threads]
    }
Client Req.
①  -> minify-prog-1 ( svc_req * , register svcxPRT *)
    {  -> svc-getarg (   )
       -> ~~Call the respective proc using fn.Ptr.~~ (Char * , SVCxPrt *)
                                                 JPEG-IN    Transport Info
    }  - Signal Thread to process all 'PROC'
                                   ②
    minify Worker ( )
    {
       -> Wait on Cond Variable / QueueEmpty ( )
       => Read Queue
       ~~-> Call Magic minify (JPEQ-IN, in.len, out.len)~~
       ~~-> svc- Send Reply ( ) ;~~            > move to prog-svc ( )
       -> minify- prog-1 - free ( );   }

minify JPEQ.h

typedef struct
{
    JPEG-IN  *IN;
    JPEG-OUT *OUT;
    struct SVCxPRT * svcTxPrt;
} minify JPEQ Request_t;  <= "Queue Item"

minify JPEQ. c
                         => Boss
minify- jpeg - prog - 1_svc ( JPEQ-~~img~~ svcxtrprt *)
{
    -> ~~Frame Queue Item~~   -> MOVE to mini-prog-1()
    -> ~~Push to Queue~~
    -> ~~Broadcast to Threads~~
}   -> Call  magic minify & return( )

minify jpeq-prog-1- free (            )
{
)
}
```



```
typedef struct {
  xdrproc_t       *xdr_argument;
  SVCXPRT         *transp;
  struct svc_req  *svcReq;
  JPEG_IN         jpeg_in_arg;
}minifyRequest_t;
```

I modified the RPC generated method signature to accommodate my new design so that I can call svc_getargs() inside minify_prog_1 () which called by RPC on client's request.

```
main (int argc, char **argv)
```

- Added a thread count options to create 't' worker threads
- minify_prog_1() with svc_register() for RPC request handler.
- Creation and Free of threads and STEQUE resources.
- svc_run ()

```
static void minify_prog_1(struct svc_req *rqstp, register SVCXPRT *transp)
```

- Initialize a function pointer and calls after parsing svc args () and
- Call initialized function pointer

```
bool_t minify_jpeg_proc_1_svc(JPEG_IN in, xdrproc_t *xdrArg, struct svc_req
*svcReq, SVCXPRT *transp)
```

- Push the svc_arg **as is** and push in to a queue and broadcast signals to threads
- Build minifyRequest_t queue item and broadcast.

```
typedef struct  {
  xdrproc_t       *xdr_argument;
  SVCXPRT         *transp;
  struct svc_req  *svcReq;
  JPEG_IN         jpeg_in_arg;
}minifyRequest_t;
```

```
MinifyWorker(void* arg)
```

- Reads from the queue
- Parse and invoke magikminify () to minify
- svc_sendreply ()
- minify_prog_1_freeresult ()

Issues and how addressed

1. Connection refused error

Slack post helped me, i had to run "sudo rpc_bind " to pass this error

2. RPC_CANTENCODEARGS error

1. I tried malloc JPEG_OUT and its members inside client -- DIDN"T work
2. Rewrote the .x with different address and generated rpcgen == DIDN"T work
3. I defined my data IN and OUT buffer in .x as

```
typedef MAX_LEN 1024
struct JPEG_IN {
 opaque buffer<MAX_LEN> ;
};
```

Suggestion: It should be infinite allocation and not as described very clearly in our class video.

3. Failure of Bonnie's "Timeout Test Case "

What caused:

I read the readme specifically about timeout and used clnt_control () in the client. But I didn't read properly about 5 seconds timeout.

How diagnosed:

I put a large timeout and then very short and found the small timeout is expected rather a longer one.

How addressed:

Later I confirmed from the readme, should have read the readme once again.

4. clnt_control () cause problem to debug MultiThreaded RPC Srv

What caused:

With my 5 seconds timeout option, I started getting timeout error when I pause at a breakpoint.

How diagnosed:

clnt_perrno() calls returned a right errors with message.

How addressed:

Increased to a very high to avoid timeouts.

References

https://www.redhat.com/archives/redhat-list/2004-June/004164.html

https://docs.oracle.com/cd/E19683-01/816-1435/rpcgenpguide-21470/index.html