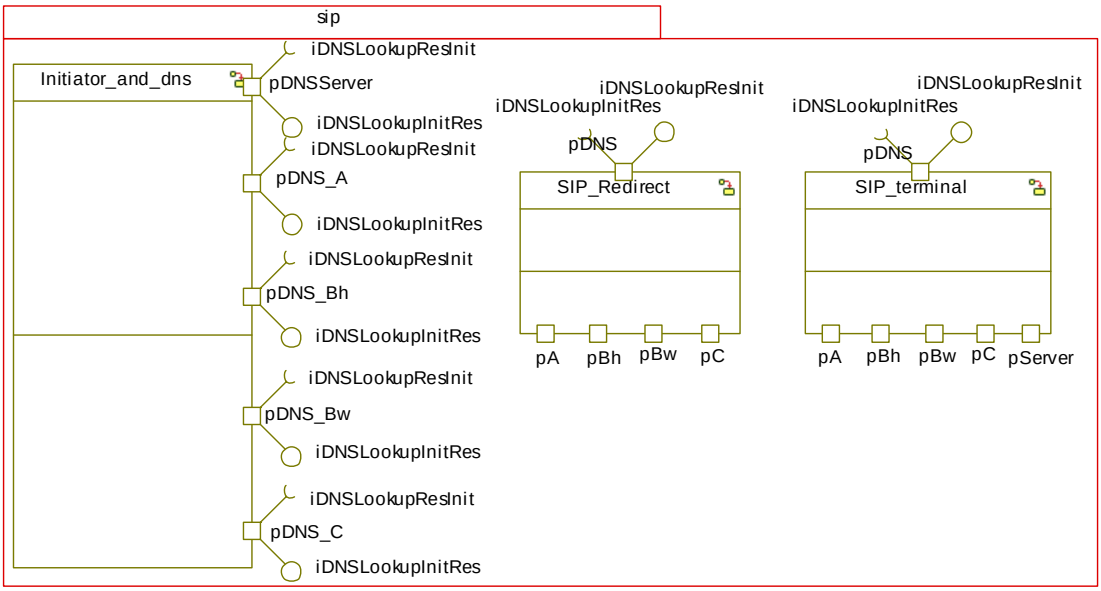
SIP Project Contents:

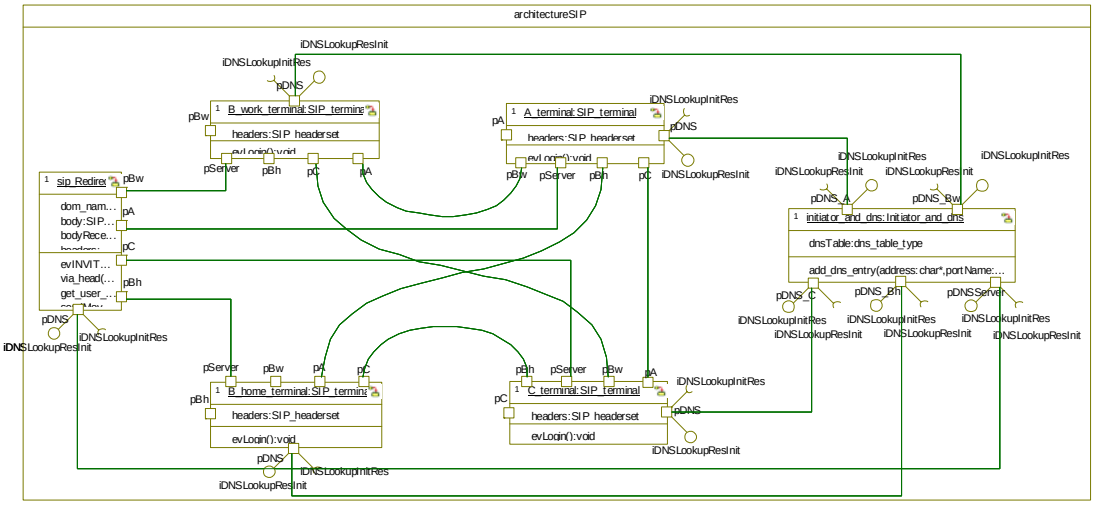
Diagrams:

Functions:

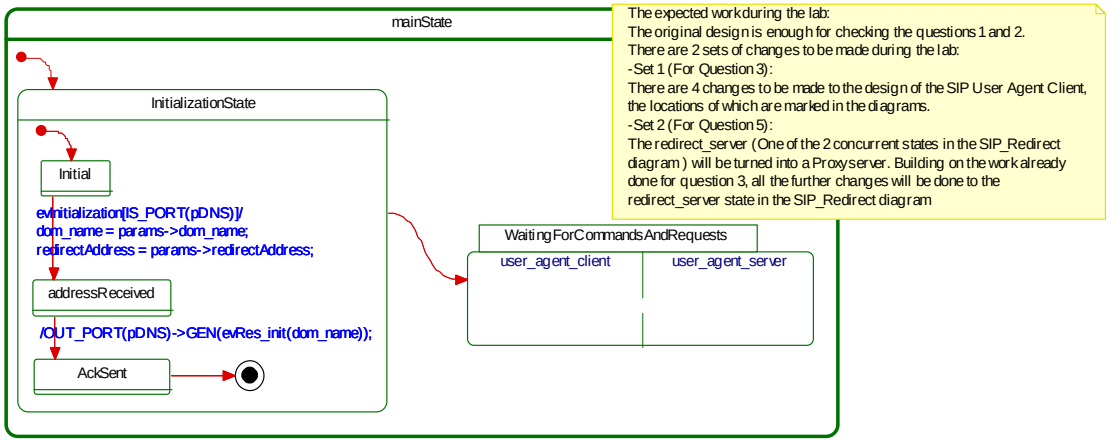
1. ObjectModelDiagram



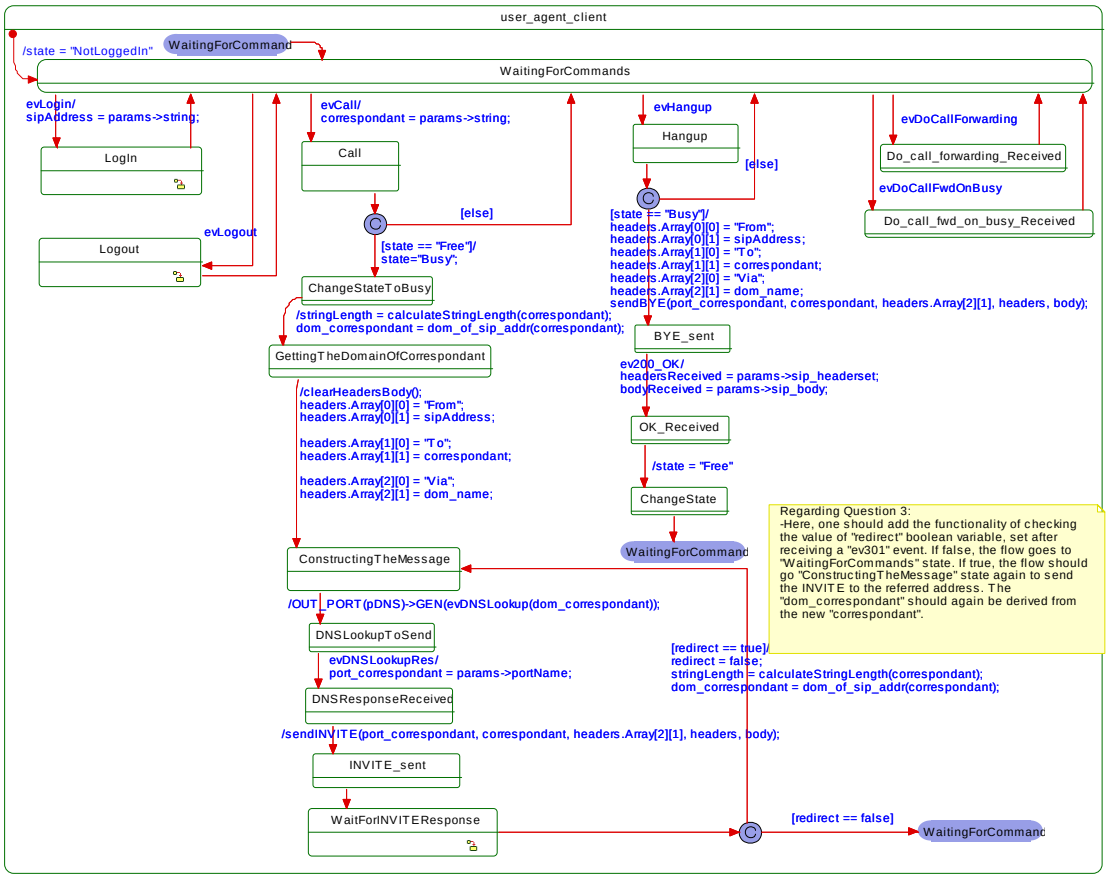
2. StructureDiagram



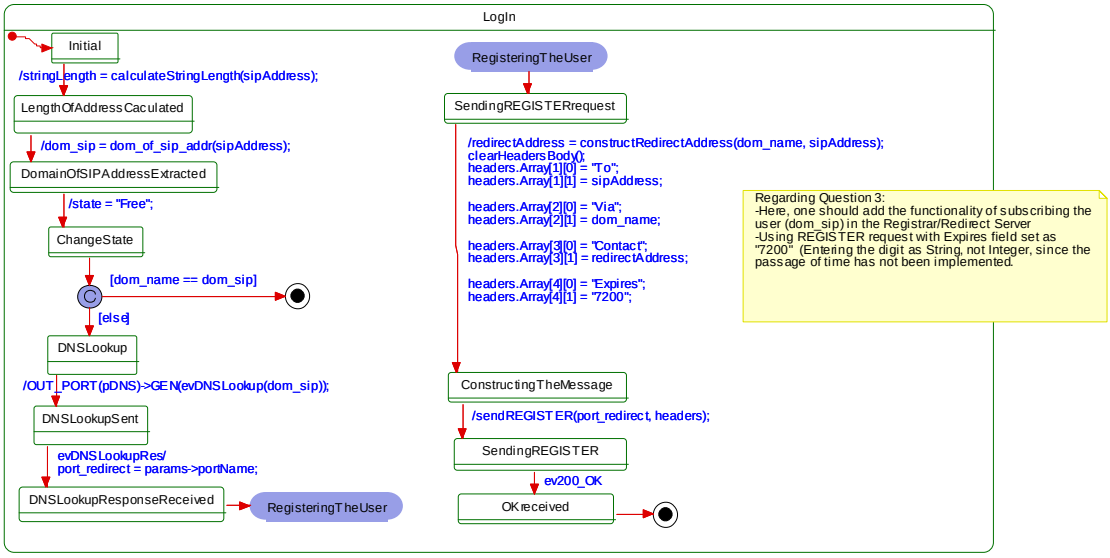
3. Statechart: SIP_terminal



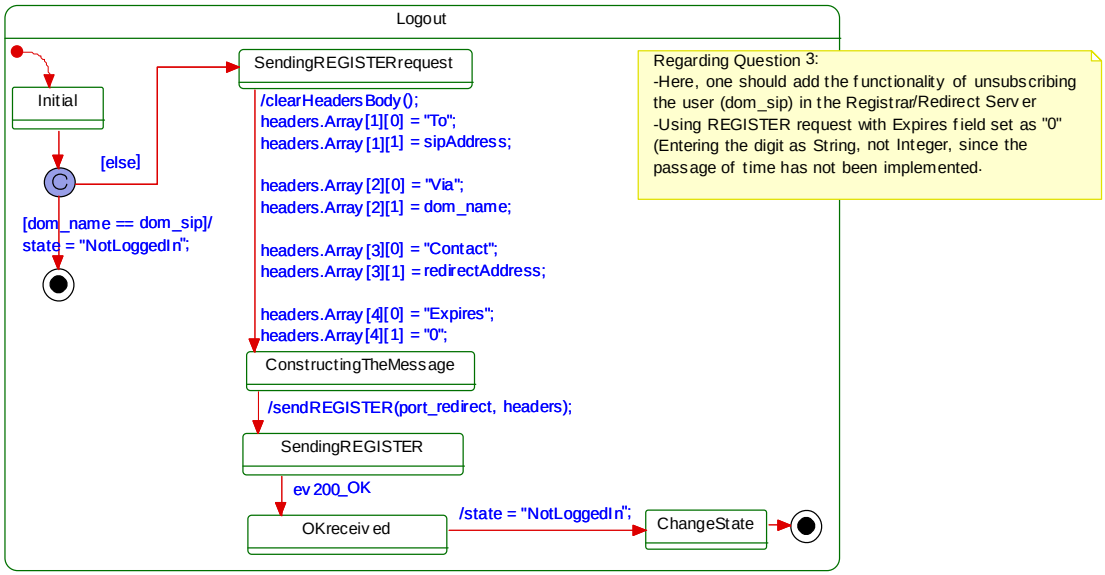
4. Statechart: user_agent_client in SIP_terminal



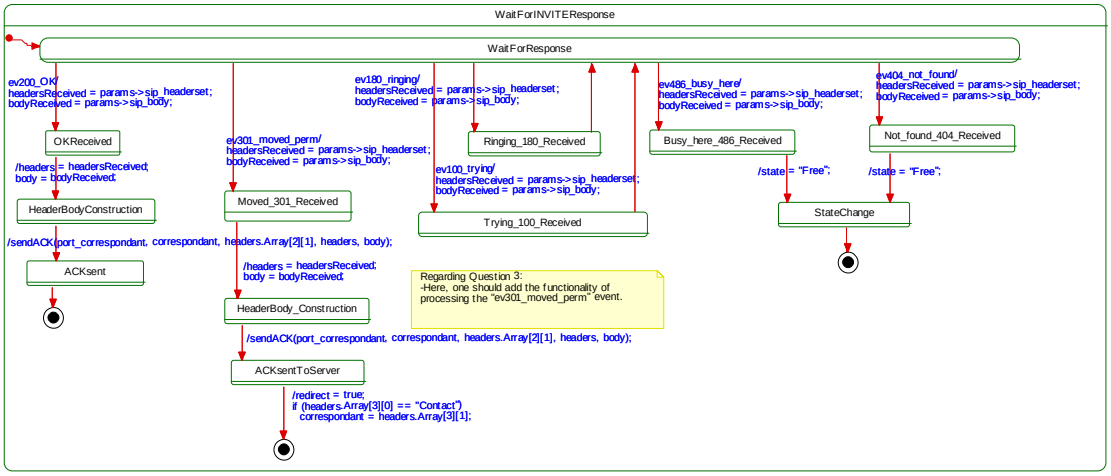
5. Statechart: LogIn in user_agent_client



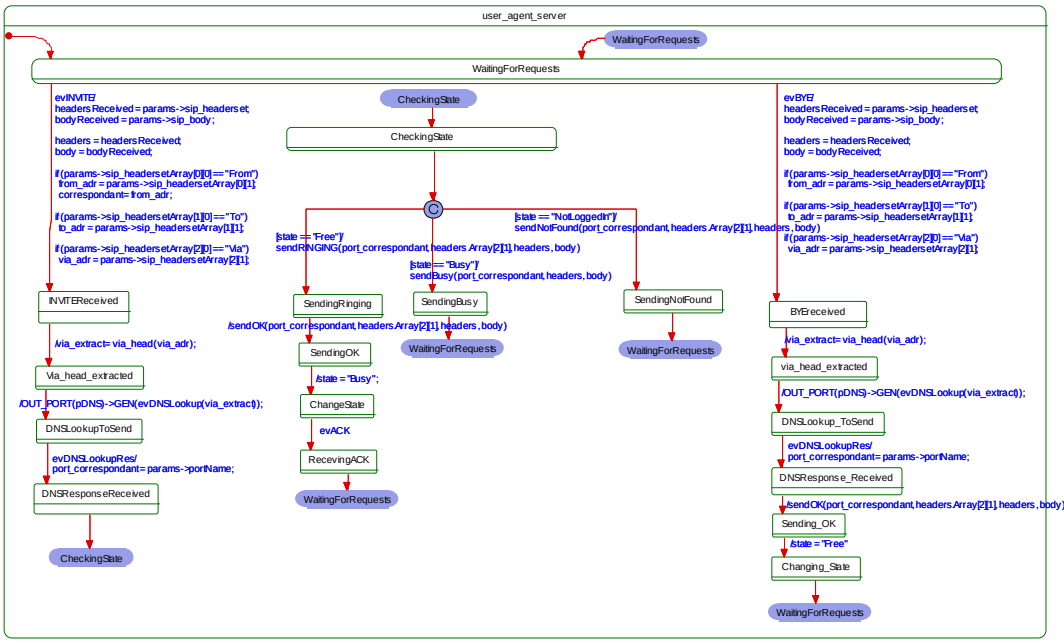
6. Statechart: LogOut in user_agent_client



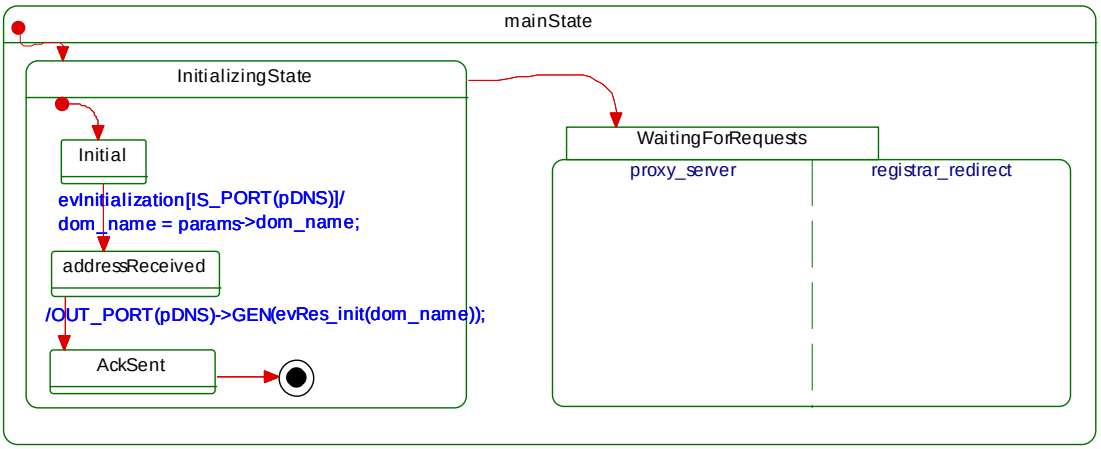
7. Statechart: WaitForINVITEResponse in user_agent_client



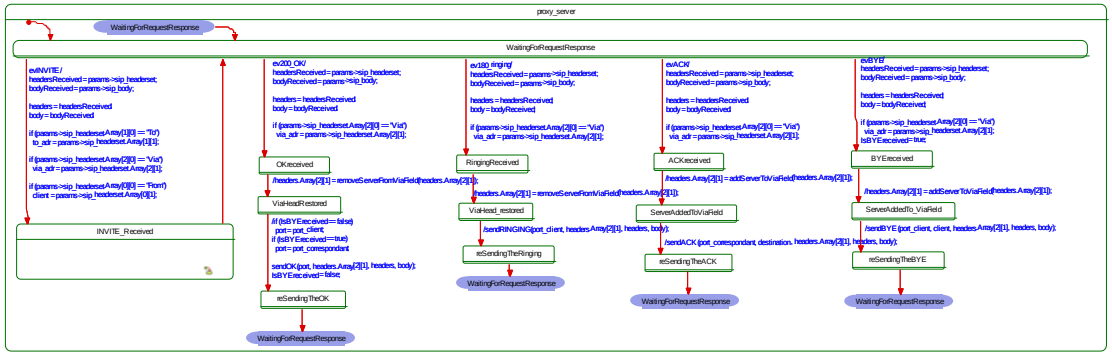
8. Statechart: user_agent_server in SIP_terminal



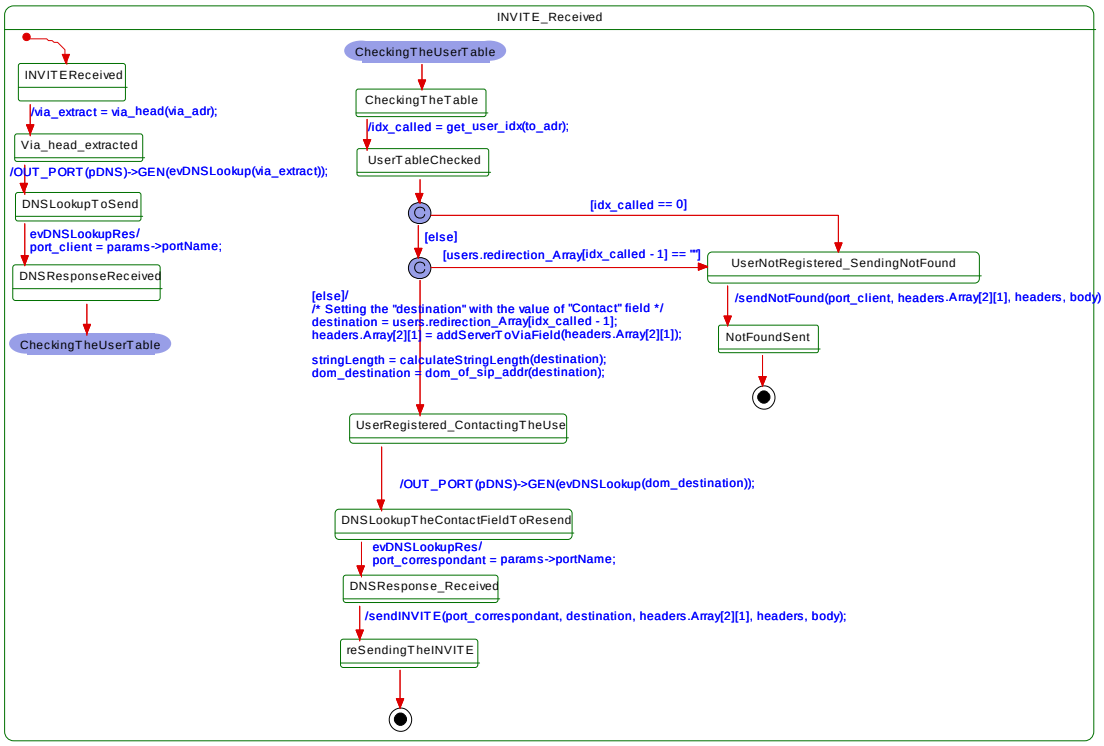
9. Statechart: SIP_Redirect



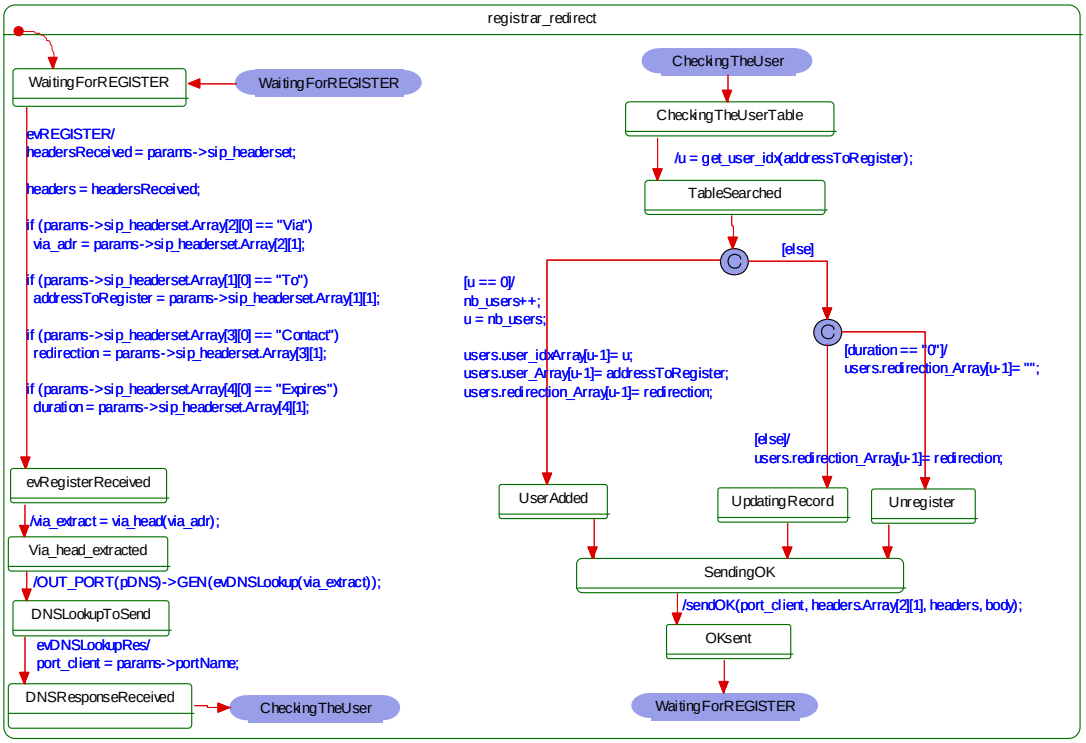
10. Statechart: proxy_server in SIP_Redirect



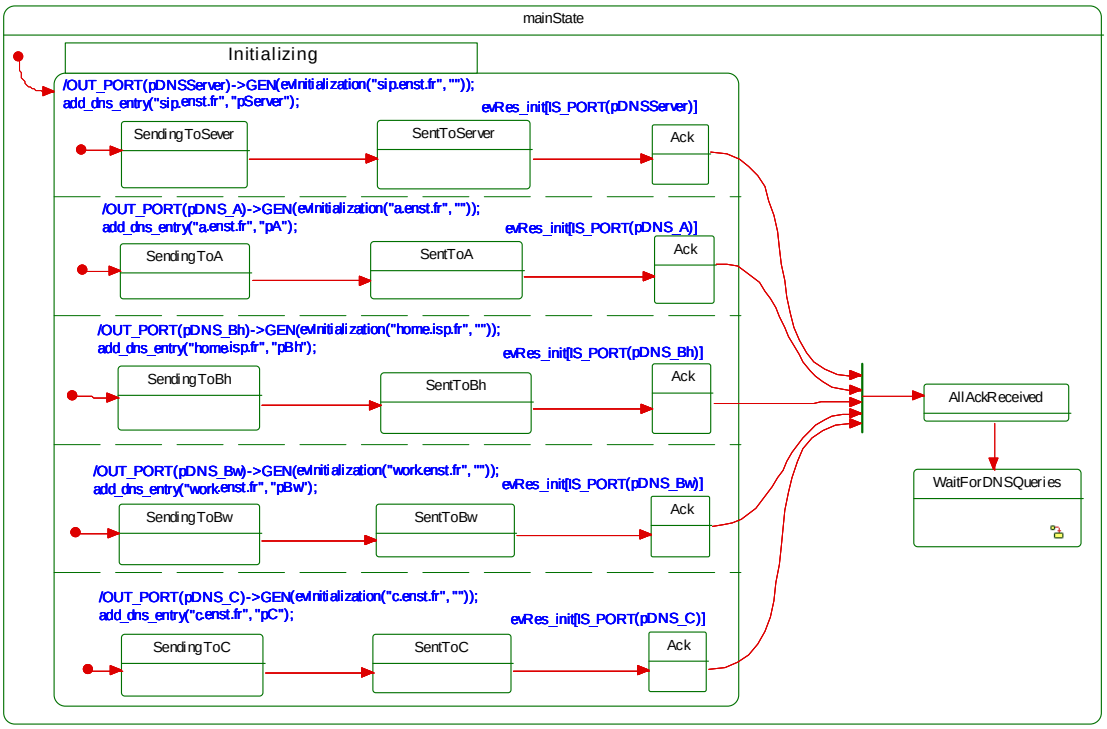
11. Statechart: INVITE_Received in proxy_server



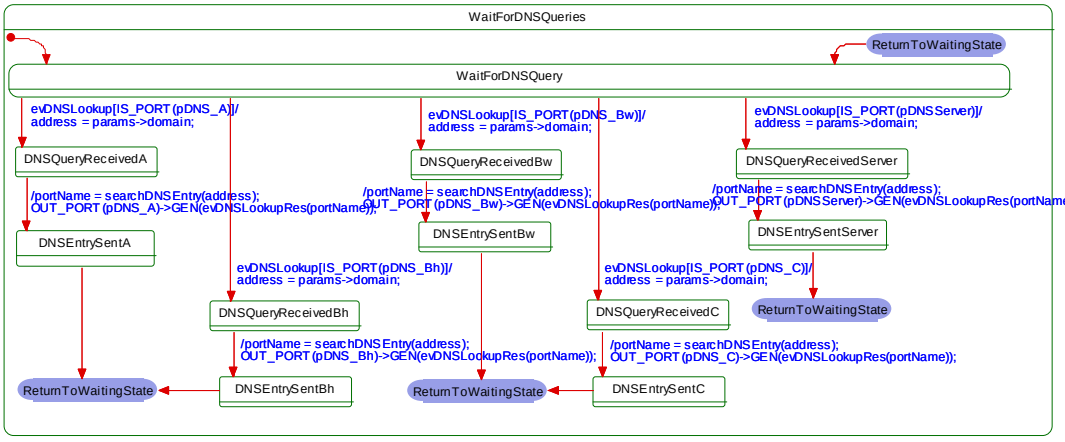
12. Statechart: registrar_redirect in SIP_Redirect



13. Statechart: Initiator_and_dns



14. Statechart: WaitForDNSQueries in Initiator_and_dns



add_dns_entry

Body:

```
for (i = 0; i < nbEntries ; i++)
{
        if (dnsTable.Array[i][0] == address)
                {
                dnsTable.Array[i][1] = portName;
                return;
                }
}

nbEntries++;
dnsTable.Array[nbEntries - 1][0] = address;
dnsTable.Array[nbEntries - 1][1] = portName;
return;
```

addServerToViaField

Body:

```
stringLength = calculateStringLength(viaHeader);

char tmpString[50];

for (i = 0; i < stringLength ; i++)
        tmpString[i]= viaHeader[i];

tmpString[stringLength]=' ';

stringLength2 = calculateStringLength(dom_name);

for (i = 0; i < stringLength2 ; i++)
        tmpString[stringLength + i + 1]= dom_name[i];

tmpString[stringLength + stringLength2 +1]='\0';

tmp_string = tmpString;

return(tmp_string);
```

calculateStringLength

Body:

```
for (i = 0 ; ; i++)
        if (string[i]== '\0')
                return i;
```

clearHeadersBody

Body:

```
for (i=0 ; i<20 ; i++)
```

```
  for (j=0 ; j<2 ; j++)
    headers.Array[i][j] = "";

for (i=0 ; i<20 ; i++)
  for (j=0 ; j<2 ; j++)
    body.Array[i][j] = "";
```

constructRedirectAddress

```
Body:
stringLength = calculateStringLength(sipAddressArg);

char tmpString[30];

for (i = 0; i < stringLength ; i++)
        if(sipAddressArg[i]=='@')
        {
                k = i;
                for(j = 0; j < i; j++)
                tmpString[j]= sipAddressArg[j];
                break;
        }

tmpString[k]='@';

stringLength = calculateStringLength(domName);

for(j = 0; j < stringLength; j++)
        tmpString[k+j+1]= domName[j];

tmpString[stringLength + k +1]='\0';

tmp_string = tmpString;

return(tmp_string);
```

dom_of_sip_addr

```
Body:
if(stringLength == 0)
tmp_string="";

char tmpString[30];

for (i = 0; i < stringLength ; i++)
        if(string[i]=='@')
        {
                for(j = i+1; j < stringLength; j++)
                tmpString[j-i-1]= string[j];

                tmpString[stringLength-i-1]='\0';

                tmp_string = tmpString;
```

```
                return(tmp_string);
        }

return(tmp_string);
```

get_user_idx

```
Body:
if (nb_users == 0)
        return 0;

for (int i =1 ; i < 11 ; i++)
{
        if (string == users.user_Array[i-1])
         return users.user_idxArray[i-1];
}

return 0;
```

removeServerFromViaField

```
Body:
stringLength = calculateStringLength(viaHead);

char tmpString[50];

for (i = stringLength; i > 0 ; i--)
        if(viaHead[i]==' ')
        {
                for(j = 0; j < i; j++)
                tmpString[j]= viaHead[j];

                tmpString[i]='\0';

                tmp_string = tmpString;
                return(tmp_string);
        }

return(viaHead);
```

searchDNSEntry

```
Body:
for (i = 0; i < nbEntries ; i++)
{
```

```
        if (dnsTable.Array[i][0] == arg)
                return(dnsTable.Array[i][1]);
}
```

sendACK

```
Body:
if (port == "pA")
        OUT_PORT(pA)->GEN(evACK(c, h, b));
if (port == "pBh")
        OUT_PORT(pBh)->GEN(evACK(c, h, b));
if (port == "pBw")
        OUT_PORT(pBw)->GEN(evACK(c, h, b));
if (port == "pC")
        OUT_PORT(pC)->GEN(evACK(c, h, b));
if (port == "pServer")
        OUT_PORT(pServer)->GEN(evACK(c, h, b));
return;
```

sendBusy

```
Body:
if (port == "pA")
        OUT_PORT(pA)->GEN(ev486_busy_here(h, b));
if (port == "pBh")
        OUT_PORT(pBh)->GEN(ev486_busy_here(h, b));
if (port == "pBw")
        OUT_PORT(pBw)->GEN(ev486_busy_here(h, b));
if (port == "pC")
        OUT_PORT(pC)->GEN(ev486_busy_here(h, b));
if (port == "pServer")
        OUT_PORT(pServer)->GEN(ev486_busy_here(h, b));

return;
```

sendBYE

```
Body:
if (port == "pA")
        OUT_PORT(pA)->GEN(evBYE(c, h, b));
if (port == "pBh")
        OUT_PORT(pBh)->GEN(evBYE(c, h, b));
if (port == "pBw")
        OUT_PORT(pBw)->GEN(evBYE(c, h, b));
if (port == "pC")
        OUT_PORT(pC)->GEN(evBYE(c, h, b));
if (port == "pServer")
```

```
        OUT_PORT(pServer)->GEN(evBYE(c, h, b));
return;
```

sendINVITE

```
Body:
if (port == "pA")
        OUT_PORT(pA)->GEN(evINVITE(c, h, b));
if (port == "pBh")
        OUT_PORT(pBh)->GEN(evINVITE(c, h, b));
if (port == "pBw")
        OUT_PORT(pBw)->GEN(evINVITE(c, h, b));
if (port == "pC")
        OUT_PORT(pC)->GEN(evINVITE(c, h, b));
if (port == "pServer")
        OUT_PORT(pServer)->GEN(evINVITE(c, h, b));
return;
```

sendMovedPerm

```
Body:
if (port == "pA")
        OUT_PORT(pA)->GEN(ev301_moved_perm(h.Array[3][1], h, b));
if (port == "pBh")
        OUT_PORT(pBh)->GEN(ev301_moved_perm(h.Array[3][1], h, b));
if (port == "pBw")
        OUT_PORT(pBw)->GEN(ev301_moved_perm(h.Array[3][1], h, b));
if (port == "pC")
        OUT_PORT(pC)->GEN(ev301_moved_perm(h.Array[3][1], h, b));

return;
```

sendNotFound

```
Body:
if (port == "pA")
        OUT_PORT(pA)->GEN(ev404_not_found(h, b));
if (port == "pBh")
        OUT_PORT(pBh)->GEN(ev404_not_found(h, b));
if (port == "pBw")
        OUT_PORT(pBw)->GEN(ev404_not_found(h, b));
if (port == "pC")
        OUT_PORT(pC)->GEN(ev404_not_found(h, b));
if (port == "pServer")
        OUT_PORT(pServer)->GEN(ev404_not_found(h, b));
return;
```

sendOK

```
Body:
if (port == "pA")
        OUT_PORT(pA)->GEN(ev200_OK(h, b));
if (port == "pBh")
        OUT_PORT(pBh)->GEN(ev200_OK(h, b));
if (port == "pBw")
        OUT_PORT(pBw)->GEN(ev200_OK(h, b));
if (port == "pC")
        OUT_PORT(pC)->GEN(ev200_OK(h, b));
if (port == "pServer")
        OUT_PORT(pServer)->GEN(ev200_OK(h, b));
return;
```

sendREGISTER

```
Body:
if (port == "pA")
        OUT_PORT(pA)->GEN(evREGISTER(sipAddress, h.Array[4][1], h));
if (port == "pBh")
        OUT_PORT(pBh)->GEN(evREGISTER(sipAddress, h.Array[4][1], h));
if (port == "pBw")
        OUT_PORT(pBw)->GEN(evREGISTER(sipAddress, h.Array[4][1], h));
if (port == "pC")
        OUT_PORT(pC)->GEN(evREGISTER(sipAddress, h.Array[4][1], h));
if (port == "pServer")
        OUT_PORT(pServer)->GEN(evREGISTER(sipAddress, h.Array[4][1], h));

return;
```

sendRINGING

```
Body:
if (port == "pA")
        OUT_PORT(pA)->GEN(ev180_ringing(h, b));
if (port == "pBh")
        OUT_PORT(pBh)->GEN(ev180_ringing(h, b));
if (port == "pBw")
        OUT_PORT(pBw)->GEN(ev180_ringing(h, b));
if (port == "pC")
        OUT_PORT(pC)->GEN(ev180_ringing(h, b));
if (port == "pServer")
        OUT_PORT(pServer)->GEN(ev180_ringing(h, b));
return;
```

via_head

```
Body:
stringLength = calculateStringLength(string);

if(stringLength == 0)
        return(string);

char tmpString[30];

for (i = stringLength; i > 0 ; i--)
        if(string[i]==' ')
        {
                for(j = i+1; j < stringLength; j++)
                tmpString[j-i-1]= string[j];

                tmpString[stringLength-i-1]='\0';

                tmp_string = tmpString;
                return(tmp_string);
        }

return(string);
```