What components have I written?

I implement class MessengerContentProvider, which hanldle content resolver's insert and query and other interfaces with SQLiteDatabase.Implement 3 test class—OnSendClickListener, OnTest1ClickListener, OnTest2ClickListener which are triggered by 3 buttons—"send","Test1" and "Test2". Another class is SequencerTotal class, which implment algorithm" Total ordering using a sequencer".

What does each component do?

Class MessengerContentProvider implment a content provider, which handle content resolver's insert and query and other interfaces with SQLiteDatabase.One design need to mention is that there are repeating insertion in test cases, so must implment content provider's insert with SQLiteDatabase. insertWithOnConflict.Class SequencerTotal provides algorithm-- " Total ordering using a sequencer". 3 test classes---OnSendClickListener,OnTest1ClickListener,OnTest2ClickListener run above class SequencerTotal.I have tried my best to separate the funtions of algorithm class(SequencerTotal) and 3 user classes. But because of the limiation of AsyncTask and the specical requirments of testcase lead to that there is still a overlap between SequencerTotal and 3 user classes.

What is my total-causal ordering algorithm? Why does it provide both ordering guarantees at the same time?

The algorithm described in figure 15.13 is very abstract, when applied to atcual, I add more detailed handling.For simplicity, it is supposed that the sequencer is one member of multicast group. I list algorithm and pseudo-code in the below. Because B_mutlicast is implemented with TCP , so it is FIFO-ordered, then the total ordering is also total-causal odering.

Algorithm for group member p

```
On initialization
        { $r_g$: := 0; $s_g$: := 0;
        intialize message queue;//store <identifer, message>
        intialize order queue;//store <identifer,S>
        }
To TO-multicast message m to group g
        {B-multicast( g, <m, i>);//i the unique identifer of the message
        }
On B-deliver(<m, i>) with g = group(m)
        {Lookup order queue with identifer "i";
         If( found and S== $r_g$)
                {
                TO-deliver m;
                Delete it from "oder queue";
                $r_g$++;
                Helper($r_g$);
                }
        else {Place <m, i> in hold-back queue;}
        }
On B-deliver( = <"order", i, S>) with g = group( )
        { Lookup "message queue" with identifer "i";
         If( found and S== $r_g$)
                {
                TO-deliver m;
                Delete it from "message queue";
                $r_g$++;
                Helper($r_g$);
                }
        else {Place <m, i> in hold-back queue;}
        if( the member is sequencer)
                {B_multicast(g,<"order",i, $s_g$ >);
                $s_g$++;
                }
        }
Helper($r_g$)
{
        Bool rg_increase=false;
        While(true)
        {
                ident=lookup "order queue" with updated $r_g$;
                if(ident found)
                {
                        msg= lookup "message queue" with "ident"
                        if(msg found)
                        {
                                TO-deliver m;
                                Delete it from "oder queue";
                                Delete it from "message queue";
                                $r_g$++;
                                rg_increase=true;
                        }
                }
        If(rg_increase==false) break;
        }
}
```

How do I handle the first message in test case 2? Why did I choose that way?

In test case 2, when receive the first message, first post the message from TCP server thread to UI thread,then create a TCP client thread to send out multicast messages. Because UI thread can't have any block operations, so I need to a new thread to send out. The reason why I don't put the "send out" action in TCP server thread is that the "send out" action don't block the TCP server thread.

What have I learned in this assignment?

When new AsyncTask to create TCP client thread , must call method execute(), can not call method executeOnExecutor() because executeOnExecutor is parallel.

I got "hands-on" distributed system expeierence from this project: the debugging, how interaction between hosts are totally different from centralized system.