Nathan Tipton

Promela model for peterson leader election protocol.
The model initialized N processes using the example init process from leader0. Each node process has a input and ouput channel. The input and output channels are assigned to create a ring of nodes with unidirectional communication. Each process is connected to its clockwise neighbor through out and counter clockwise neighbor via in. There are three possible states for a process: Active, Relay, and Stop. In order to avoid invalid end states we use the end keyword to label relay as an end state. All processes start as active and has a unique identity or process number. When a process is active it initially sends it's own id which it assumes is the max. The process then waits to receive the value *e* from its neighbor. If e does not equal d then it also waits for the second message *f*. Next the process compares d,e,and f. If e is the max of the three values d is reassigned to the value of e otherwise the process goes to relay.
Processes in relay wait to receive input d, check if d is their identity. If it is not they pass on d to the output.

2. Verify the following properties using the SPIN model checker:
(a) There is always at most one leader. **PASS**
(b) Eventually always a leader will be elected. **PASS**
(c) The elected leader will be the process with the highest number. **PASS**
(d) The maximum total amount of messages sent in order to elect a leader is at most 2Nblog2Nc + N. **PASS**

```
/*Peterson leader election protocol
Nathan Tipton A01207112


*/
#define N      6      /* nr of processes */
#define I      3      /* node given the smallest number    */
#define L      12     /* size of buffer  (>= 2*N) */
#define X      38     /* 2*N*(Log2(N))+6*/
ltl p1 {<>[] (nr_leaders==1)};
ltl p2 {[] (messages<X)};
ltl p3 {[] (nr_leaders <= 1)};
ltl p4 {[] (leader_id==N)};
```

```
chan q[N] = [L] of {byte};

//counter for total leaders elected
byte nr_leaders = 0;

//counter for total messages sent
byte messages=0;
//used to check if leader has the largest ID which should be N
byte leader_id=N;

proctype node (chan in, out; byte ident)
{    byte d,e,f;

//Set exclusive read and write access for chan for this process
       xr in;
       xs out;

//     printf("MSC: %d\n", ident);
//Initially output my identity which I assume is the max until told otherwise

 activ:
//     printf(“active[%d]\n”,ident);
     d=ident;
     do :: true -> out!d;
          messages++;
          in?e;
          if :: (e==ident) ->
               //process is the leader
               goto stop
            :: else -> skip
          fi;
          if::d>e->out!d;
           ::else-> out!e;
          fi
          messages++;
          in?f;
          if :: (f==ident) ->
          //process is the leader
               goto stop
            ::else ->skip
          fi
```

```
        if::(e>=d)&&(e>=f) -> d =e
         ::else -> goto relay //put process into passive mode
        fi
    od;

relay:
//relay mode
//printf("RelayNode[%d]\n",ident);
end:
  do :: in?d->
      if::(d != ident)->
      //process is not the leader
        out!d
        messages++;
       ::else -> goto stop //process is the leader
      fi
  od;

stop:
printf("Leader: %d",ident);
nr_leaders++;
leader_id=ident;
  skip
}


/* initialize N processes with IDs*/
init {
      byte proc;
      atomic {
            proc = 1;
            do
            :: proc <= N ->
                  run node (q[proc-1], q[proc%N], (N+I-proc)%N+1);
                  proc++
            :: proc > N ->
                  break
            od
      }
}
```