One-Lane Bridge Example

```
Monitor Bridge {
   int nN = 0, nS = 0;
   cond oktogosouth, oktogonorth;

   procedure go_north_request() {
      while (nS > 0) wait(oktogonorth);
      nN = nN + 1;
   }

   procedure go_north_done() {
      nN = nN - 1;
      if (nN == 0) signal_all(oktogosouth);
   }


   procedure go_south_request() {
      while (nN > 0) wait(oktogosouth);
      nS = nS + 1;
   }

   procedure go_south_done() {
      nS = nS - 1;
      if (nS == 0) signal_all(oktogosouth);
   }
}
```

To convert the above monitor to a server process using asynchronous message passing, we first need to define the required types and channels.

```
type op_kind = enum(GOSOUTHREQ, GOSOUTHDONE, GONORTHREQ, GONORTHDONE);
chan request(int clientID, op_kind kind);
chan reply[n]();
```

Note that a client process / a car process should work as follows:

```
process NorthBoundCar[i = 0 to k-1]
   while (true) {
        send request(i, GONORTHREQ);    // non-blocking
        receive reply[i]();             // blocking

        cross the bridge …

        send request(i, GONORTHDONE);   // non-blocking
        receive reply[i];               // blocking
   }
}
```

The definition of the `SouthBoundCar` process is similar.

```
process BridgeServer {
   int nN = 0, nS = 0;
   int clientID, op_kind kind;
   queue northpending, southpending;

   while (true) {
        receive request(clientID, kind);
        switch(kind) {
         case GONORTHREQ:
           if (nS > 0) insert(northpending, clientID);
           else {
                nN = nN + 1;
                send reply[clientID]();
           }
           break;

         case GONORTHDONE:
           nN = nN - 1;
           send reply[clientID]();
           if (nN == 0) {
                while (!empty(southpending)) {
                   remove(southpending, clientID);
                   nS = nS + 1;
                   send reply[cliendID]();
                }
           }
           break;

         case GOSOUTHREQ:
           …
         case GOSOUTHDONE:
           …
   }
}
```

Notes:

1. The last northbound car passing the bridge is responsible for notifying all waiting/delayed southbound car.
2. In the case of `GONORTHDONE`, `remove(southpending, clientID)` removes the front item of the `southpending` queue and assign it to variable `ClientID`, so the variable `clientID` now contains the ID of a southbound car. Note that `ClientID` previously contains the ID of a northbound car.
3. The ID of a southbound car is inserted to the `southpending` queue in the case of `GOSOUTHREQ`, when condition `(nN > 0)` is true.