```
type op_kind = enum(ACQUIRE, RELEASE);
chan request(int clientID, op_kind kind, int unitid);
chan reply[n](int unitID);

process Allocator {
  int avail = MAXUNITS; set units = initial values;
  queue pending;  # initially empty
  int clientID, unitID; op_kind kind;
  declarations of other local variables;
  while (true) {
    receive request(clientID, kind, unitID);
    if (kind == ACQUIRE) {
       if (avail > 0) {  # honor request now
          avail--; remove(units, unitID);
          send reply[clientID](unitID);
       } else   # remember request
          insert(pending, clientID);
    } else {   # kind == RELEASE
       if empty(pending) {  # return unitID to units
          avail++; insert(units, unitid);
       } else {  # allocate unitID to a waiting client
          remove(pending, clientID);
          send reply[clientID](unitID);
       }
    }
  }
}

process Client[i = 0 to n-1] {
  int unitID;
  send request(i, ACQUIRE, 0)    # "call" request
  receive reply[i](unitID);
  # use resource unitID, then release it
  send request(i, RELEASE, unitID);
   ...
}
```

Figure 7.7 Resource allocator and clients.