```
type howInvoked = enum(CALL, SEND);
type howServiced = enum(PROC, IN);
type opRef = rec(howServiced how; int machine, opid);

proc invoke(howInvoked how; opRef op; byte values[*]) {
  if (how == CALL)
    insert executing on call delay list;
  if (op.machine is local)
    localInvoke(executing, how, op, values);
  else {    # machine is remote
    netWrite(machine, INVOKE, (executing,how,op,values));
    dispatcher();
} }

proc localInvoke(int caller; howInvoked inv;
                 opRef op; byte values[*]) {
  if (op.how == PROC) {
    get free process descriptor;
    if (inv == CALL)
      save identity of caller in the descriptor;
    else    # inv == SEND
      record that there is no caller (set caller field to zero);
    set program counter for the process to op.address;
    push values onto process stack; insert descriptor on ready list;
  }
  else {    # op.how == IN
    look up class descriptor for the operation;
    if (inv == CALL)
      append(opclass, caller, op.opid, values);
    else    # inv == SEND
      append(opclass, 0, op.opid, values);
  }
  dispatcher();
}

proc append(int opclass, caller, opid; byte values[*]) {
  if (opclass is locked) {
    insert (caller, opid, values) into new invocations list;
    move processes (if any) from wait list to access list;
  }
  else {   # opclass not locked
    insert (caller, opid, values) into pending list;
    if (wait list not empty) {
      move first process to ready list;
      move other processes to access list;
      set the lock;
} } }
```

Figure 10.11 Invocation primitives.