```
declarations of network buffers, free descriptors, delay list

netRead_handler: {  # entered with interrupts inhibited
  save state of executingP;
  acquire new buffer; prepare network controller for next read;
  unpack first field of input message to determine kind;
  if (kind == CALL)
    handleRPC(caller, address, value arguments);
  else  # kind == RETURN
    handleReturn(caller, results);
}

proc rpc(int machine, address; byte args[*]) {
  netWrite(machine, CALL, (executing,address,args));
  insert descriptor of executing on delay list;
  dispatcher();
}

proc handle_rpc(int caller, address; byte args[*]) {
  acquire free process descriptor; save identity of caller in it;
  put address in a register for the process;
  unpack args and push them onto the stack of the process;
  insert process descriptor on ready list;
  dispatcher();
}

proc rpcReturn(byte results[*]) {
  retrieve identity of caller from descriptor of executing;
  netWrite(caller's machine, RETURN, (caller, results));
  put descriptor of executing back on free descriptor list;
  dispatcher();
}

proc handleReturn(int caller; byte results[*]) {
  remove descriptor of caller from delay list;
  put results on caller's stack;
  insert descriptor of caller on ready list;
  dispatcher();
}
```

Figure 10.9 Kernel routines for implementing RPC.