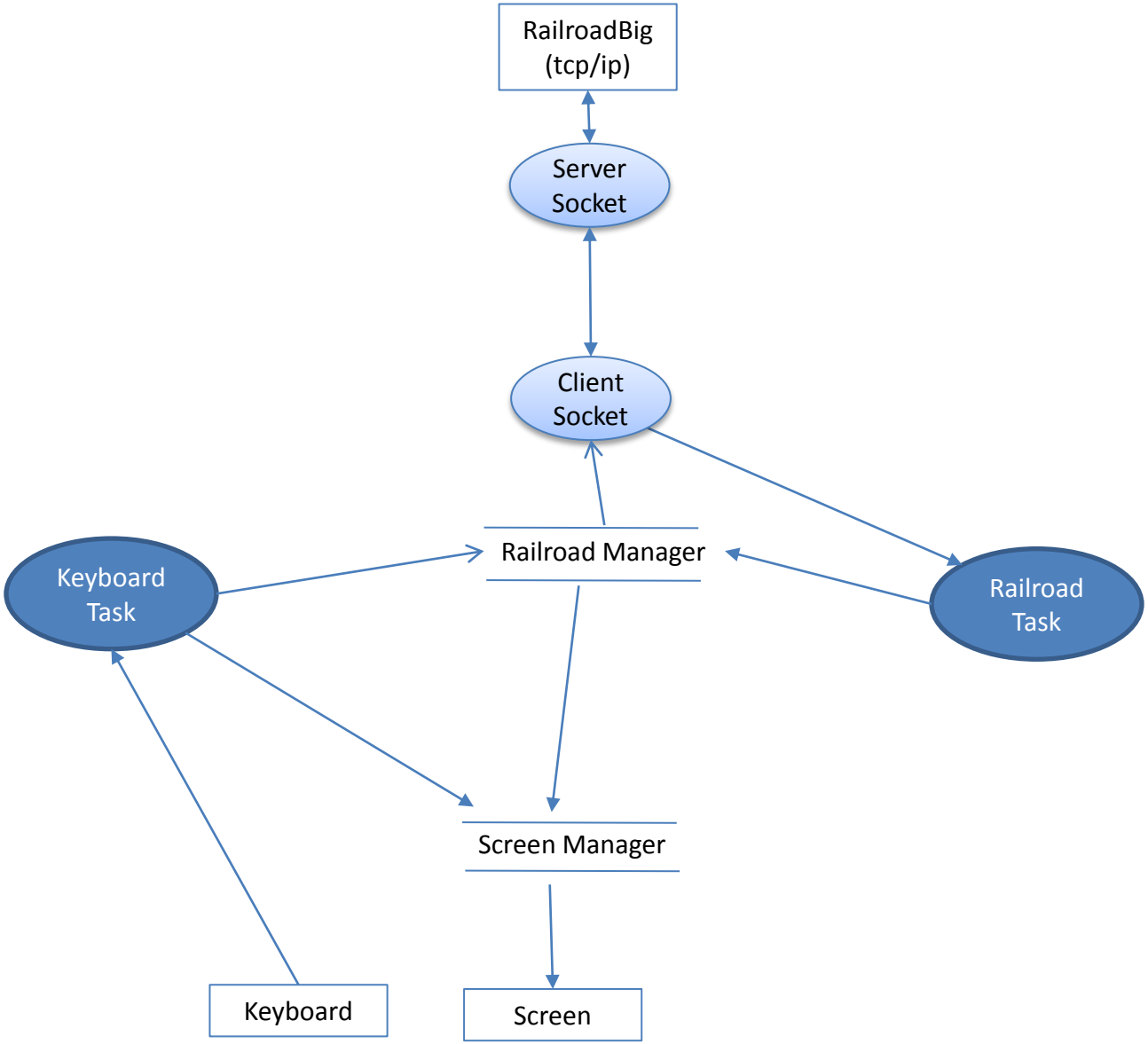
Admin Throttle

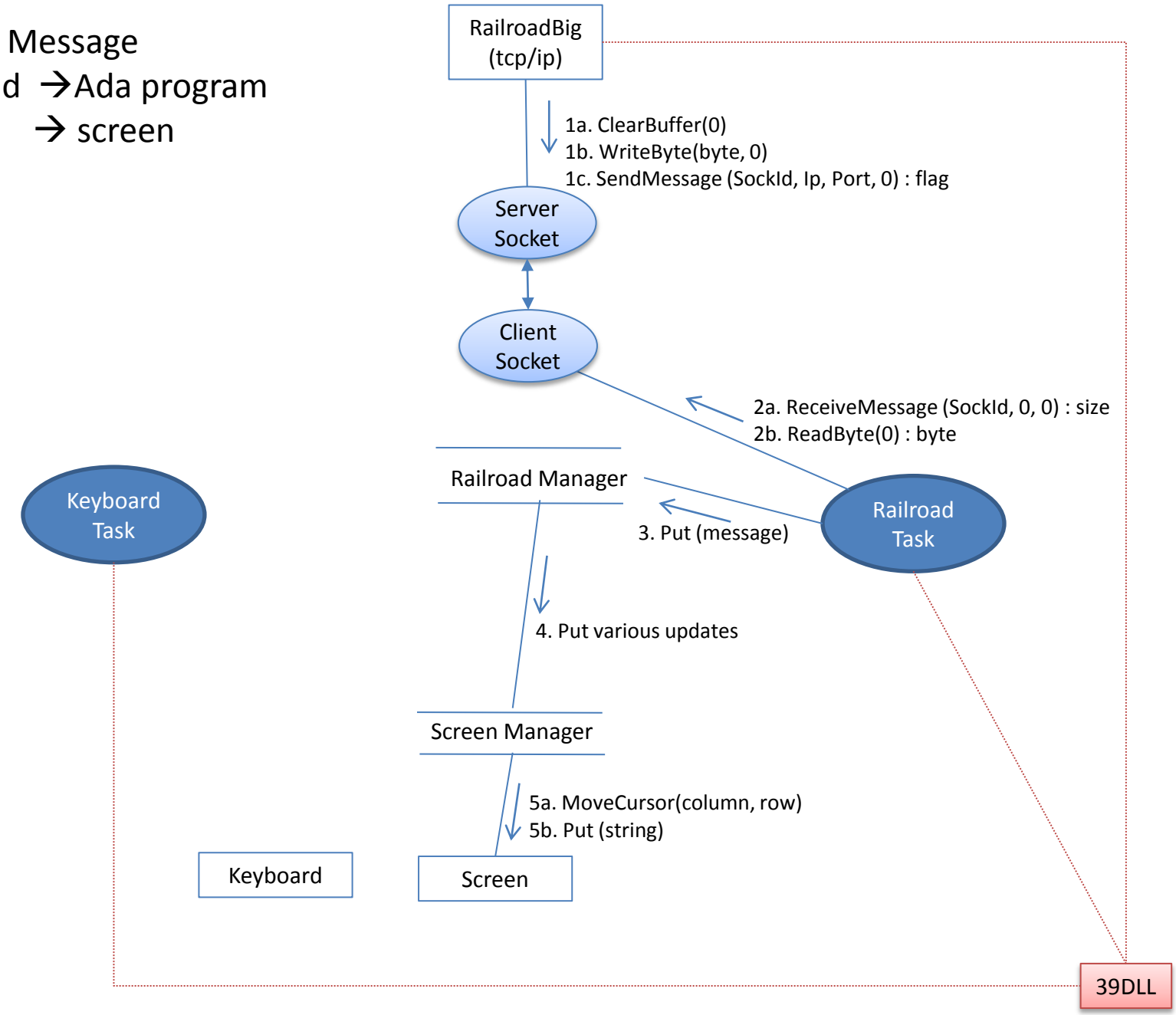
Design

Admin Throttle

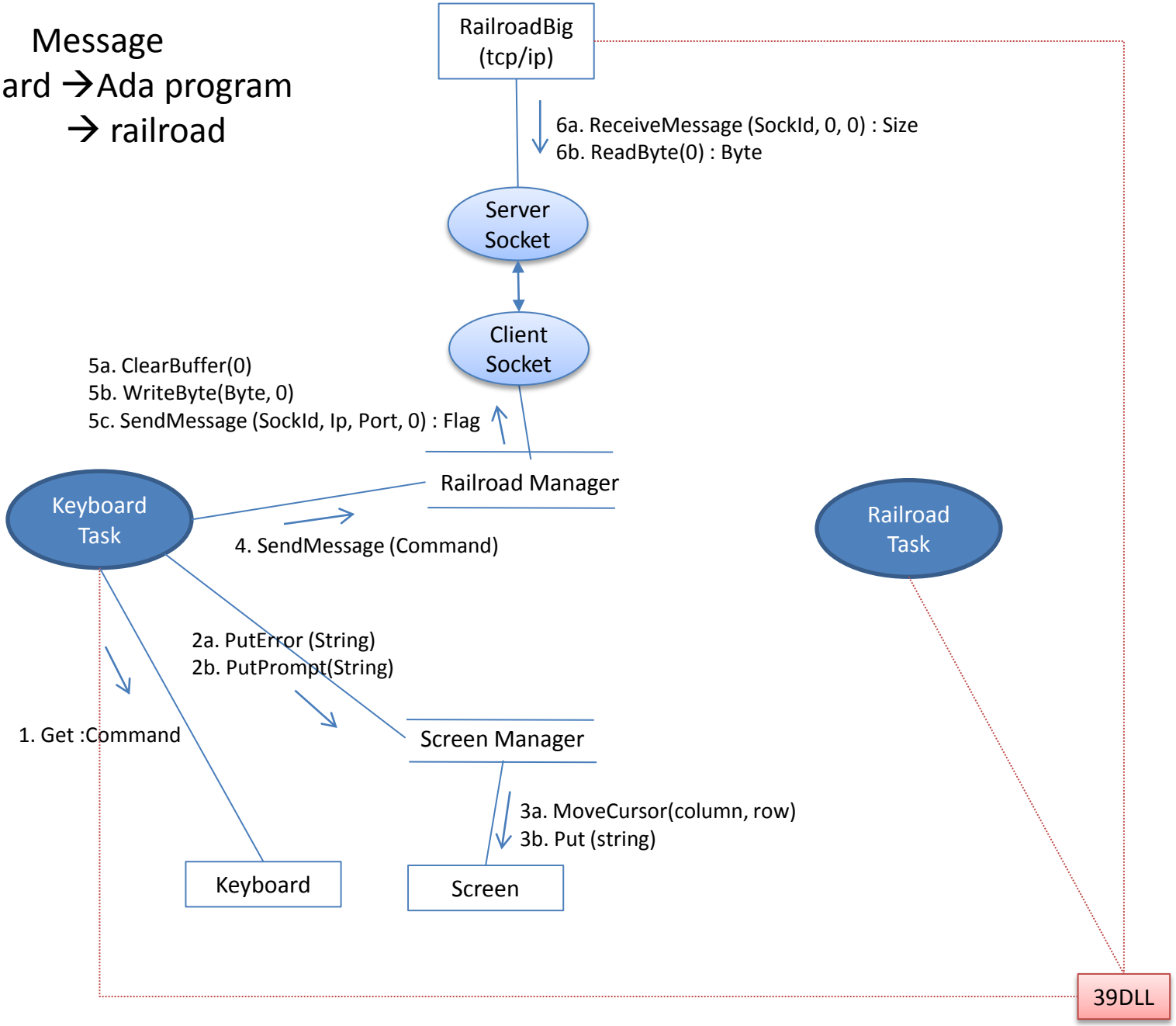
```
          1         2         3         4         5         6         7 . . . to      100
    12345678901234567890123456789012345678901234567890123456789012345678901234567890. . .

01  Train PhyAdr/Slot VirAdr/Slot State Speed Dir Light Bell Horn Mute Location
02  ----- ----------- ----------- ----- ----- --- ----- ---- ---- ---- --------
03    1    3333 / 3      13 / 10    W    60    F   on   off  off  off  13   14   26
04    2    2222 / 2      14 / 11    BH   0     R   on   on   on   on   103  104  77
05    -
06    -
07
08  Last msg received: set speed train 2 to 50
09                     bytes in decimal / bytes in hex
10  Switches                          Commands
11    0 1 2 3 4 5 6 7 8 9             Xf  read XML      Za  s#…s# initialize loco
12  0   T T T T T T C C T             SEa select loco   STa steal loco
13  1 C c T C T C C T C t
14  2 T T C C C C C                   Cs close          Ts throw
15                                    Vt n -- set velocity of t to n
16                                    Ft forward        Rt reverse       Q  quit
17                                    Bt bell           Ht horn          .  halt all
18                                    Lt light          Mt mute
19                                    P+ power on       P- power off
20                           a = physical loco address  s = switch       s# = sensor#
21                           t = train                  f = file name    n  = 0..127
22  Error messages displayed here
23  >
```


RailroadBig
(tcp/ip)
Server
Socket
Client
Socket
Railroad Manager
Keyboard
Task
Railroad
Task
Screen Manager
Keyboard
Screen

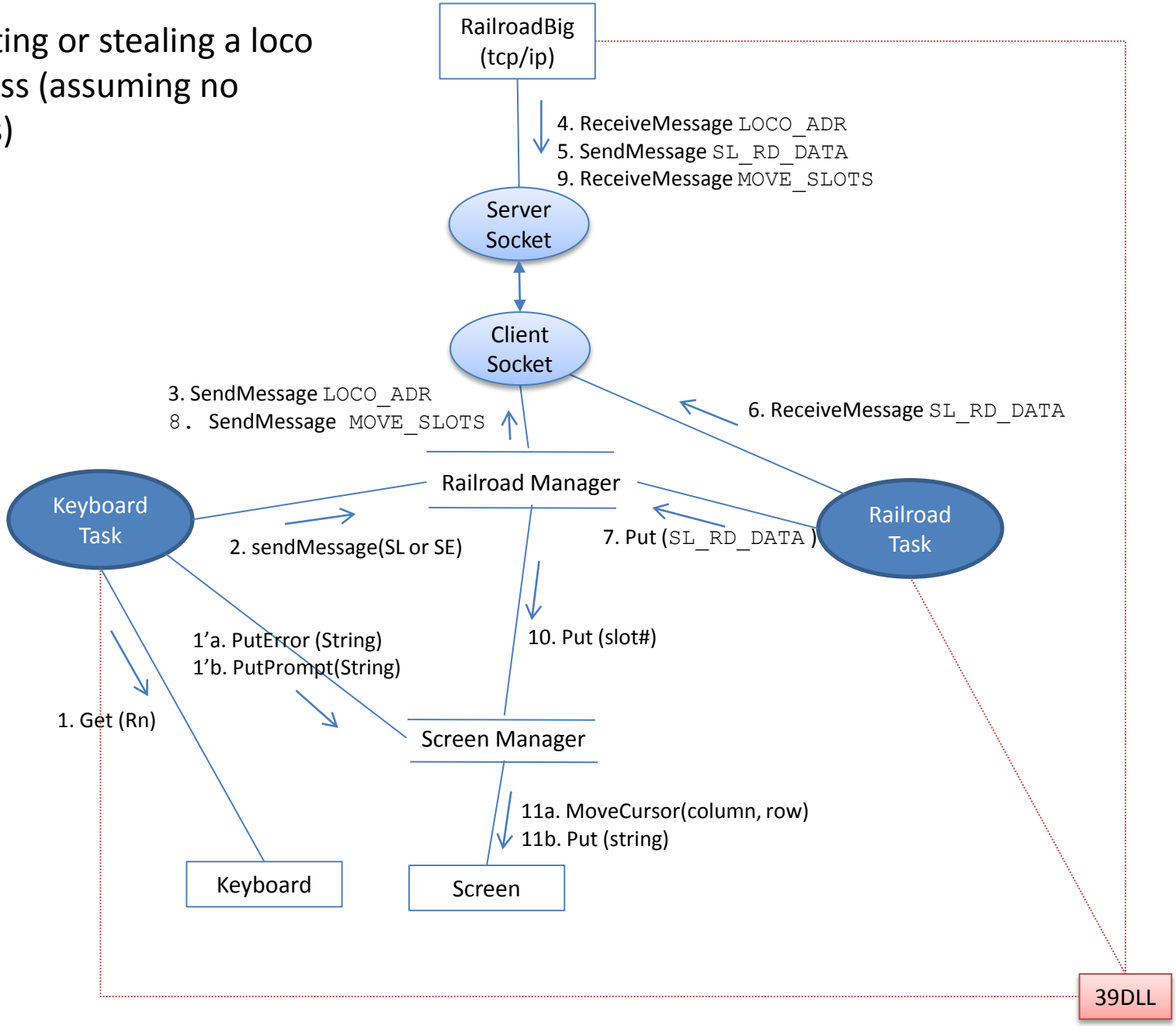
Message
railroad →Ada program
→ screen



Message
keyboard → Ada program
→ railroad



Selecting or stealing a loco address (assuming no errors)



Packages

Globals.ads

```
+ kNumTrains          := 4
+ kNumSwitches        := 26
+ kMaxLenMsg          := 14
+ kMaxLenError        := 80
+ kGreen              := "green"
+ kWhite              := "white"
+ kRed                := "red"
+ kYellow             := "yellow"
+ kNoColor            := "      "

+ kDeltaSpeed         := 10

+ ColorType           string(1..6)
+ LocoAddressType     (0..4444)
+ SlotType            (0..120)
+ DirectionType       (Forward, Backward)
+ SpeedType           (0..127)
+ OnOffType          (On, Off)
+ TrainType          record of
                        isConnected,
                        phyadr, physlot, viradr,  virslot, state, direction,
                        speed, light, horn, bell, mute, sensors(a list of natural)
+ TrainArrayType     (1..kNumTrains) of TrainType

+ SwitchType         (Closed, Thrown, Moving, Unknown)
+ SwitchArrayType    (1..kNumSwitches) of SwitchType

+ ByteArrayType      array (Integer RANGE <>) of unsigned_8
+ MessageType        bytes(1..kMaxLenMsg), InUse

+ CmdType            (ReadXML, InitializeLoco, SelectLoco, StealLoco, Close, Throw, Forward, Backward,
                      Increase, Decrease, Halt, Horn, Bell, Mute, Light, Power, MoveSlots, Unknown, Quit)
+ CommandType        record of cmd, natural
```

Globals.ads

```
+    ConnectSocket      C.Double;
+    IpStrAda           String := "127.0.0.1";
+    IpStrC             Chars_Ptr := New_String(IpStrAda);
+    CEmptyString       Chars_Ptr := New_String("");
+    ServerPort         c.double := c.double(14804);
+    Buffer0            C.Double := C.Double(0.0);
+    BlockingMode       C.double := c.double(0);
+    NonblockingMode    C.double := c.double(1);
+    CValue             C.Double;
+    CZero              C.double := c.double(0);
```

Packages

Api39dll
+f dllInit
+f TcpConnect
+f ClearBuffer
+f WriteByte
+f SendMessage
+f ReceiveMessage
+f ReadByte

ScreenManager
+p Initialize
+p PutTrains(Trains)
+p PutMessage(Stgring)
+p PutSwitches(Switches)
+p PutError(String)
+p PutPrompt(String)

+o objScreenManager

RailroadTask
+e Start

+o obj RailroadTask

KeyboardTask
+e Start

+o objKeyboardTask

Screen
+d ScreenDepth
+d ScreenWidth

+p Beep
+p ClearScreen
+p MoveCursor

RailroadManager
-d Trains
-d Switches
-d Last message received

+p Initialize
+p Put(Message)
+f GetMessage(Command, Success):Message

+o objRailroadManager

AdminThrottle

It all starts here
Connect to server
Start tasks
Initialize managers

At startup

- ❑Trains initialized to
  - slot number for a train is 0 until registered
  - direction forward, speed 0
  - light, horn, bell, mute all off
- ❑Message sent to throw all switches
- ❑Last message received is blank

```
-- Constants for direction, lights, horn, bell, mute, turnout action
kForward      := 16#00#;
kBackward     := 16#20#;  -- 0010 0000
kLightsOn     := 16#10#;  -- 0001 0000
kLightsOff    := 16#00#;
kHornOn       := 16#02#;  -- 0000 0010
kHornOff      := 16#00#;
kBellOn       := 16#01#;  -- 0000 0001
kBellOff      := 16#00#;
kMuteOn       := 16#08#;  -- 0000 1000
kMuteOff      := 16#00#;

-- Constants for switches
kCloseIt      := 16#20#;  -- 0010 0000
kIsClosed     := 16#10#;  -- 0001 0000
kThrown       := 16#00#;  -- 0000 0000

-- Opcodes
OPC_GPON       := 16#83#; -- power on
OPC_GPOFF      := 16#82#; -- power off
OPC_INPUT_REP  := 16#B2#; -- report sensor fired
OPC_SW_REP     := 16#B1#; -- report turnout now open/thrown

OPC_LOCO_SPD   := 16#A0#; -- set speed
OPC_LOCO_DIRF  := 16#A1#; -- set direction, horn, bell, lights
OPC_LOCO_SND   := 16#A2#; -- set mute and unmute sound
OPC_SW_REQ     := 16#B0#; -- move a turnout

OPC_LOCO_ADR   := 16#BF#; -- request for slot data
OPC_SL_RD_DATA := 16#E7#; -- slot data response
OPC_LONG_ACK   := 16#B4#; -- insufficient slots
OPC_MOVE_SLOTS := 16#BA#; -- register slot
```

All the above are declared as “constant unsigned_8”

Messages Sent

OPC_SW_REQ – move switch
<0xB0><SW1><SW2><CHK>
  byte1 := switch number (0..127)
  byte2 := switch direction

OPC_LOCO_SPD – set speed
<0x A0 ><SLOT#><SPEED><CHK>
 byte1 := slot;
 byte2 := speed

OPC_LOCO_DIRF – set direction, lights, horn, bell
<0xA1><SLOT#><DIR_STATE><CHK>
 byte1 := slot;
 byte2 := 16#00# or Direction or Light or Horn or Bell;

OPC_LOCO_SND – set mute
<0xA2><SLOT#><SOUND><CHK>
 byte1 := slot;
 byte2 := 16#00# | Mute;

OPC_LOCO_ADR – request information about a loco address
<0xBF><adrhigh><adrlow><chk>
  byte2 := 16#FF# and (LocoAddress mod 128);
  byte1 := 16#FF# and ((LocoAddress - byte2) / 128);

OPC_MOVE_SLOTS – register a slot
<0xBA><slot#><slot#><chk>
 byte1 := slot;
 byte2 := slot;

checksum := 16#FF# xor opCode xor byte1 xor byte2;
clearbuffer();
writebyte(opCode);
writebyte(byte1);
writebyte(byte2);
writebyte(checksum);
sendmessage(socketToSimulator);

Messages Received

All the output messages, which are displayed on the screen, but otherwise ignored.

OPC_INPUT_REP – report sensor fired

<0xB2><SN1><SN2><CHK>

SN1 <0,A6,A5,A4,A3,A2,A1,A0>

SN2 <0,X,I,L,A10,A9,A8,A7>

I = 1 => add 1000 to the address

L = 1 => sensor high else low

OPC_SW_REP – report turnout now open/thrown

<0xB1><SW1><SW2><CHK>

SW1 <0,A6,A5,A4,A3,A2,A1,A0>

SW2 <0,1,I,L,A10,A9,A8,A7>

I = 1 => turnout closed else thrown

OPC_SL_RD_DATA – report loco address information

<0xE7><0E><slot#><status><adrlow><spd><dirf><trk><ss2><adrhigh><snd><id1><id2><chk>

0 1 2 3 4 5 6 7 8 9 10 11 12 13

slot# the number of the slot

status address already registered in the indicated slot (D5D4 == 11) or
the slot is available for registering the address (D5D4 == other)
(The simulator makes no attempt to get the other bits correct and
sets them all to 0's.)

address that was sent

other information that we choose to ignore so we set it to 0x00;

OPC_LONG_ACK – report that there aren't enough slots

<0xB4><lopc><ack1><chk>

lopc 0x00, indicating insufficient slots

other information that we choose to ignore so the simulator sets it to 0x00;