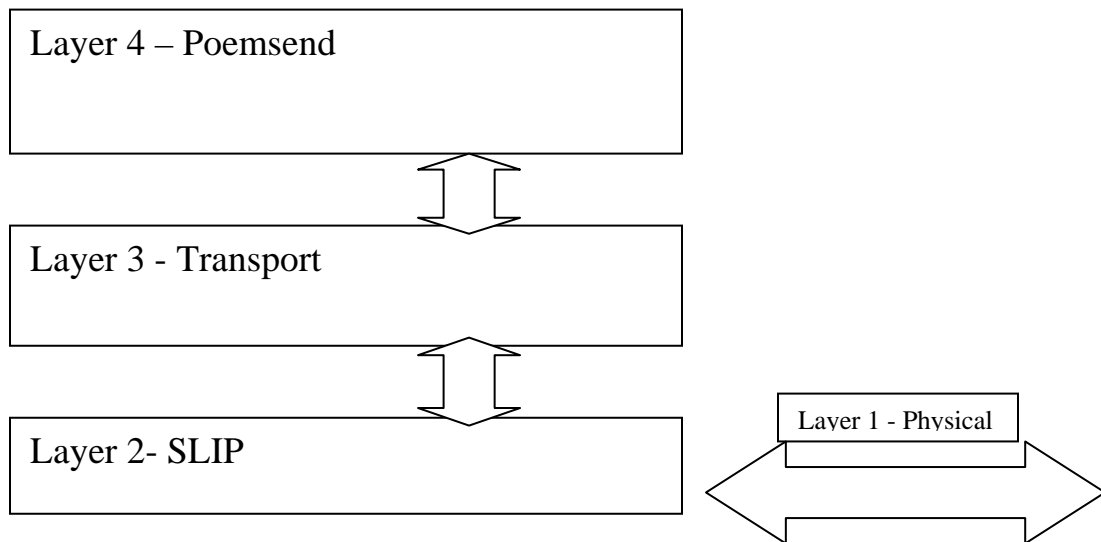


Gur Doitel

## MESSAGE PASSING PROTOCOL STACK

The Message Passing Protocol Stack is a three layered model designed to allow the development of higher level applications necessitating message passing between two z8 boards. Excluding the first layer, Physical Layer, the three layers are: Layer 2, the Data Link Layer, is an implementation of SLIP or Serial Line over IP Protocol RFC1055; Layer 3 the Transport Layer, much like the OSI layer 4 transport layer; and a Layer 4, Application Layer, “Poem Send” which in this case is a simple exchange of high level messages between the two boards.



### LAYER 2- Data Link Layer - SLIP

SLIP is a very simple protocol, basically sending and receiving one byte at a time over the UART.

The SLIP protocol defines two special characters: END and ESC. END is octal 300 (decimal 192) and ESC is octal 333 (decimal 219) .

The sending procedure gets a datagram, a char[], and for every data byte checks to see if the bytes is the same as a special character which denotes the end of transmission. If it is, another special character is sent right after to let the receiver know that transmission will not be ending and to keep receiving. When the last byte is sent, the END character is sent.

### Deficiencies of Slip

- addressing: SLIP currently provides no mechanism for hosts to communicate addressing information over a SLIP, this doesn't matter to this application since it is only meant for two boards directly connected over serial line or IR.

- type identification: SLIP has no type field. Thus, only one protocol can be run over a SLIP connection.

- error detection/correction: noisy phone lines will corrupt packets in transit. Slip will not detect or correct errors since it is simply too slow. Error correction and Detection is left up to the higher levels.

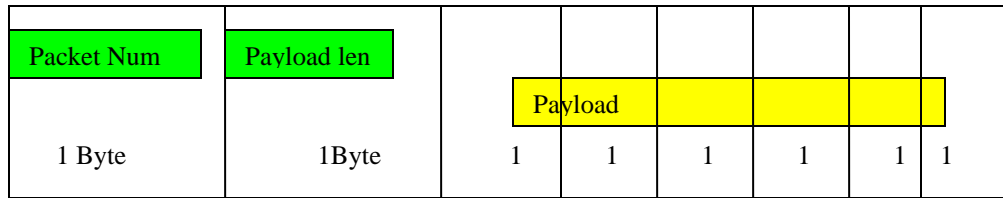
### LAYER 3 Transport Layer

The transport layer takes high level messages from the Application layer and places them into smaller sized packets or Datagrams. Each packet has a Header and a payload.

The Header – a 2 byte header, the first byte represents the packet number, from the total number of packets sent to complete the message to one. A packet with the number one is the last packet, and therefore denotes the end of transmission. The second byte of the header represents the number of bytes in the payload.

The Payload – the payload is a maximum 6 bytes. A 6 byte MTU provides for very slow throughput, but the large number of packets it creates would make for a more interesting demonstration, which for this project made sense. The payload is calculated by getting the length of the message with “strlen(message)”, and doing a mod 6, to find if there is a remainder. If there is a remainder then that remainder is the payload of the last packet, otherwise all packets have a 6 byte payload.

LAYER 3 DATAGRAM



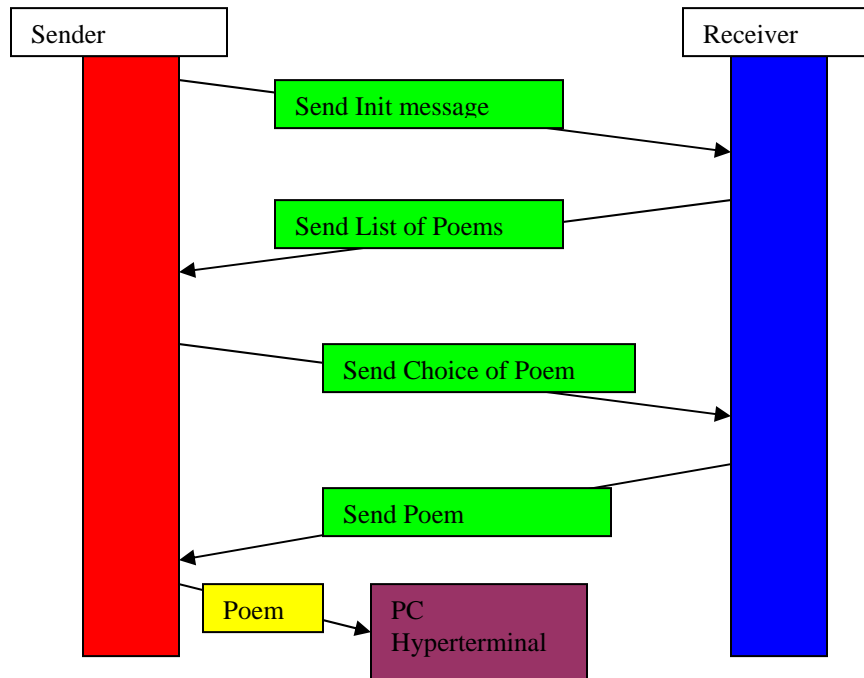
Error Detection

Layer 3 provides Error detection through its Packet number tracking. When a packet number is extracted, if it is more than one integer smaller than the previous, then a packet was missed, and an error is detected. Error correction is not provided, but could be by sending a special message at the end of transmission stating which packets were lost if any, those packets could then be retransmitted.

LAYER 4 – Application Layer – Poem send

The application layer is closely tied to the interface provided by the board. The boards provide three buttons, sw1, sw2, and sw3. When the application is initialized, if sw1 is pressed, this signifies that that board is the initiator or client, or I will call them sender. If sw3 is pressed, then the board acts as server, but I call them receiver. Both send and receive, so tis terminology may be a little confusing. This is a blocking protocol, a sender and receiver reverse role at each exchange, and rely on the response of their counterpart to advance.

Sender- “initiator” The sender immediately sends an “initiation” message to the receiver, then goes into a receive mode, where it sits and waits. Upon receipt of the response, a “list” of available poems, the sender polls the buttons for the choice of the user. The button pressed represents the choice, and a message is constructed using that choice and sent too the receiver. The sender the waits for another response, the response that arrives is the poem, which is immediately sent to the pc running hyperterminal to display the poem.



### Project Difficulties

The Project is very software heavy, and required an extreme amount of debugging. I used hyperterminal to debug, by sending all my messages to the pc, this proved to be a very unreliable method. The issue of reliability of hyperterminal really distorted the state of the application. I could not really tell whether a message was not received or if the terminal was just not displaying as it would often do seemingly without reason. I could not have anticipated the amount of debugging it would require to get this project working properly. This experience definitely exposes the difficulties of developing in this environment.

The project had to be altered at the last minute, when difficulties arose from designing the application for heterogeneous boards, with different interfaces. After receiving a second z8 board, many of the problems were averted through the ability to write a symmetric program, but not enough time was left to debug.

The connection of the boards to each other, and to the PC proved to also be a challenge, especially when having to constantly load code onto the boards then switch the cable from one port to the other. This resulted in a broken pin on the serial cable which I have

replaced.

