Data Network Milestone 3 protocol design

Haiyang Xu, Zhe Fang, Hao Ji

September 19, 2011

1 Layer and protocol

In milestone 3, we have to solve the following three problems:

1. Low MTU;
2. Frame corruption/loss;
3. Multiple nodes;

In the case of Low MTU, we can transplant our solution of milestone 2 to this scenario in milestone 3, which is dividing a large frame into pieces such that satisfying the condition of Low MTU.

And in the case of Frame corruption and loss, we have designed a timeout-resend mechanism. Every time a node send a frame to a link, it will wait for the ACK of that frame for a round trip time. If it doesn't receive the ACK from the destination node (because of frame loss or corruption), it will automatically resend that frame.

And in the case of Multiple nodes which is also the most complex situation, we use the flooding algorithm. In each node we will implement a minimal network layer where a routing table will be maintained. This routing table contains following fields:

```
typedef struct {
    CnetAddr address; // ... of remote node
  int ackexpected; // packet sequence numbers to/from node
  int nextpackettosend;
  int packetexpected;
  int minhops; // minimum known hops to remote node
  int minhop_link; // link via which minhops path observed
} NLTABLE;
```

during every cycle the node maintain this table. e.g. to address 12, we can query the expected ack from that address node, next packet to send, expected packet, minimum hops and minimum hop link. And at the beginning, no minimum hop link is known to a certain address. So we will just broadcast the message to every link. And after a while it will converge to an equilibrium in which a stable min hops and min hop link map to a certain address.

We decided to divide the system into 2 layers. A simple data link layer and a simple network layer. The datalink layer is mainly responsible for writing message to the physical layer. And the network layer is responsible for the routing among multiple nodes.

2 Packet header format

```
typedef struct {
  CnetAddr src;
  CnetAddr dest;
  NL_PACKETKIND kind; /* only ever NL_DATA or NL_ACK */
  int seqno; /* 0, 1, 2, ... */
  int hopcount;
    int checksum;
    size_t pieceNumber;
    int pieceEnd;
  size_t length; /* the length of the msg portion only */
  char msg[MAX_MESSAGE_SIZE];
} NL_PACKET;
```

As illustrated above, a packet header contains source and destination IP address. Since a message will be seperated into several small subpacekts during transmission, each subpacket contains a *seqno*, which indicates the sequence number of a ceratin subpacket in the whole message. The *hopcount* shows how many hops will be traveled during current transmission. The *pieceEnd* indicates whether current subpackect is the last subpackect of the whole message.

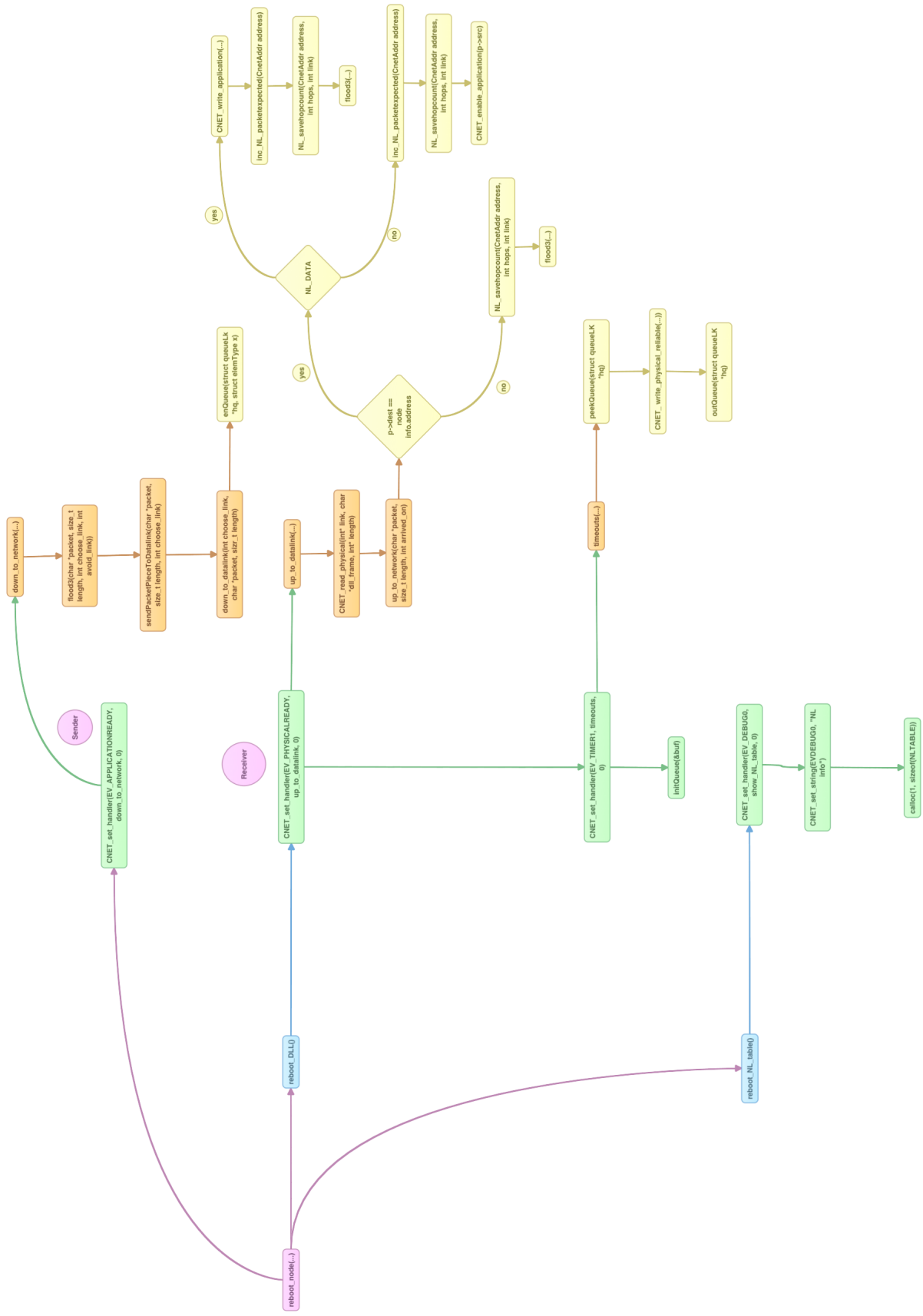

Figure 1: milestone 3