```
Network Working Group                                           CS 640
Request for Comments: rfc-echimp-draft                    Section 0001
                                                             Spring'05

Obsoletes RFC:                                              April 2005
```

e-CHIMP is the CS 640 Hierarchical IP-based Messaging Protocol (e-CHIMP)

-=-

Status of this Memo

This memo is the official specification of e-CHIMP for CS 640 Spring '05.
Distribution of this memo is unlimited.

1. INTRODUCTION

The e-CHIMP protocol specifies the interactions between the different
entities and provides a teleconferencing mechanism between users. It is
run on different machines in a distributed fashion. In its basic form
it is a text-based protocol in which users can interactively
communicate with other users connected to the e-CHIMP network consisting
of a single server.

2. OVERVIEW

There are two main entities : Server (S) and Clients (C).
Clients are users of the system, while server provides the appropriate
support for communication between clients.
Each client in the system is connected to a specific server, which
we call the Designated Server for that client.

Each client and server in the system is assigned a unique ID.
There is a one-to-one mapping between <IP-addr,port>
pair to which the client/server is bound to, and the ID.

2.1 Client Initialization:

A client enters the system it needs to choose a nickname for itself.
It does this by contacting the server as described later. It can
then find out about all clients connected to the server, by performing
an appropriate 'who' query at the server. The server responds with
a list of nicknames of currently joined clients, their IDs and
also their IP address, port number information.
(This information may be stale at the server, and hence inaccurate
sometimes.)
The client subsequently can send and receive messages with any of
the other clients. Additionally each client can join one or more
groups and exchange messages on that group.

2.2 Group Structure :

A group is a set of clients that want to communicate on a specified
topic. It is equivalent to a channel of an IRC. Each group has a
'group leader' -- which is one of the clients that
is part of the group. The group leader maintains information about all
the members of the group, and disseminates this to other new members
joining the group.
The group leader will send periodic heart-beat messages to all members
(clients) of the group to announce its presence once every 3 seconds.
The heart-beat message contains a list of all currently known group
members.

When a client wants to join the group, it first finds out

who the group leader is (this it does from the Designated Server, as described later) using the GROUP_LEADER_REQUEST message.
It then informs the group leader its intent to join (JOIN_GROUP). The group leader will accept all such requests from non-joined members. These messages need no acknowledgments. Clients would need to handle loss of these messages, e.g. by appropriate re-transmissions after timeouts. We recommend a timeout period of 3 seconds.

When a client, A, leaves the group, it would try to inform the group leader if possible (using LEAVE_GROUP). It is also possible that no message could be sent, e.g. if the client fails. When a client silently leaves, it may go undetected for a while. However, user data is sent to all the clients of the group reliably, and eventually some client will detect the missing client (since no ack for its data will be received).
In this protocol, three consecutively missed acknowledgments will imply the sudden departure of a client, and appropriate mechanisms to purge this state from the group leader should be implemented.

2.3 Data exchanges :

All data exchange must be reliable but should be implemented using UDP (as done in PA 3). The source client is responsible for distributing data reliably to the destination client(s).

There are two kinds of data messages :
- One to one or private messages
- One to many or group messages

All data messages are exchanged between clients directly.

For private messages, the destination client will send an ack to the source client indicating proper receipt of the message.

For group messages, one such ack is expected from each of the destination clients. The message is sent as a replicated set of unicast messages originating at the source client. Retransmissions should be directed to each of the destination clients that did not ack.

Since each client is informed of all the other members of a group, by periodic heart-beats from the group leader, it knows exactly which clients to send the messages. However, it is possible that some of the destination clients may have silently left the group without being able to inform the group leader. In this case, the message to that client will not elicit acks as described before.

2.4 Group leader management :

It is also possible that a group leader fails, in which case the group would be temporarily without a group leader. In this case, the heart-beats from the group leader will stop and all the members will realize that the group leader is gone. In this case, all members will initiate the process of choosing a new group leader. The implementers of the protocol can define independent mechanisms to perform new leader election. This RFC specifies one way to do this, as follows:
The leader of a group sends periodic heart-beat messages to its Designated Server. The heart-beat period is 3 seconds. If the server does not receive three consecutive heart-beats from the leader, it assumes that the leader has left the group.
Additionally each member of the group (say C) will also detect the failure of the group leader. On detecting such a failure, it will use a random timer (between 0 and 30 seconds) to send a new GROUP_LEADER_REQUEST message to the Designated Server. The server will choose the first client to be the new leader and inform the client of this choice using GROUP_LEADER_RESPONSE message.

The chosen client then informs all other members of the group
about the change in group leadership. Note that it is possible
that some clients do not correctly receive any or all of these
messages. Under such circumstances the affected client will stop
getting messages sent to the group. Hence the only recourse to
this client is to rejoin the group (JOIN_GROUP).

2.5 Names --- nick names and group names:

Each client in the system has a unique nick name which identifies
the user. Similarly each group in the system has a unique name
which identifies the group.
To ensure that the nick names and the group names are unique,
all assigned nick names and group names are stored at the Designated
Server. Each client checks with the Designated Server the
availability of a name before allowing its use. The first client to
claim a name is given the name and the Designated Server retains
a copy of the name to check against duplication.
A client can change its nick name, thereby, releasing its control
on the prior name at the server. A new client can subsequently request
a released nick name.

Each client also periodically (with a period of 3 seconds) sends
a heart-beat message to the server. If three consecutive heart-beats
are not received by the server, then the latter assumes that the
client has silently departed from the group. In such cases, the
server releases the nick name which was being used by the client.

Nick names and group names are both between 1 and 12 characters in
length.

2.6 Designated Server:

The Designated Server maintains a database (but you do
not have to implement all the complexity of a database, any simple
structured file will serve the purpose). It stores three things :
- List of all alloted nick names and client to which it is attached
- List of all group names
- For each group alloted, the id of the client which serves as the
leader for that group.

When a new group is created, typically a client will request a
new group by sending a GROUP_LEADER_REQUEST message.
If the Designated Server has not seen this group name before,
it will accept the group name and set the client as the group
leader. It will respond with a GROUP_LEADER_RESPONSE message
as appropriate.

Similarly when a new nick name is requested by a client with the
NICK_REQUEST message, the Designated Server will respond with a
NICK_RESPONSE message to either accept the nick name if it is
unique or reject if it is not.

The formats of these messages are described later in the messages
part.

3. MESSAGES

All messages between clients and the Designated Server have the following
format:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |Version|        Msg Type       |     Src Id    |     Dest Id   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |             Length            |            Checksum           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Rest of the message (depends on the message type) ...       |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Version string : 4 bits
Msg Type : 12 bits
Src Id : 8 bits
Dst Id : 8 bits
Length : 16 bits
Checksum : 16 bits

The first 8 bytes is called the Outer Header (similar to the idea of an IP header that encapsulates the data part).

The Src and Dst Id are the Ids of the originating and destination client or server.

The Length field indicates the Length in bytes, of the message, and does not include the first 8 bytes.
The Checksum is computed over the outer header only.
This field stores the 1’s complement of the 1’s complement sum of the header.

3.1 General Messages (outer header) :

There are 4 message types :
0,1: (RESERVED FOR FUTURE USE).

2 : (UNRELIABLE DATA) : These are unreliable messages between clients and Designated Server.

The contents of the DATA part is ASCII text data that is interpreted by the client as appropriate.

3 : (RELIABLE DATA) :

This will be implemented using the reliable TCP protocol that you implemented in the prior programming assignment.

The contents of the DATA part is as shown below :

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   Byte Sequence Number                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Acknowledgment                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Time stamp (seconds)                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   Timestamp (microseconds)                    |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        Length                         |S|F|A|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
   |       All zeros               |           Checksum            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
   |                                                               |
   |                              Data                             |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

For named groups of clients, e-CHIMP uses a multi-unicast scheme to send messages.

Messages reliability for data transfer is discussed in the client to
client reliability section.

3.2 Message details:

A number of different messages need to be exchanged between clients and
the Designated Server in the e-CHIMP network. In this specification we outline some
of these messages, that can serve as a guideline for implementing
the protocol.

3.2.1 Name Management Messages:

These are sent using the UNRELIABLE DATA message format of the outer
header generic message type.
The following messages are defined :

- NICK_REQUEST : Client requests a nick name.
  The data part of the message contains :

  ```
  NICKREQ<sp><requested nick name upto 12 bytes><sp><Q/U><sp><Y/N>
  ```

  Q : Query from a client as to which client has this nickname.
  U : Request this nickname to be reserved for this client.

  Y: If this message is a heart-beat.
  N: Otherwise.

  The first time a client sends a nick name request to a server,
  it should await a positive response from the server before
  installing and using the nick name.
  Subsequently, the client will send the nick name request to the
  server periodically (and will serve as a heart-beat). If the
  nick name request is a heart-beat, the server will check if the
  nick name specified in the heart-beat is the correct nick name for
  the client. If this is true, the server will not need to respond
  to the heart-beat.
  If the server finds that the nick name does not match, it will
  send a negative nick name response to the client (described next).

- NICK_RESPONSE : Response to above
  The data part of the message contains :

  ```
  NICKRESP<sp><requested nick name echoed back><sp>
      <3 byte client-ID in ASCII><sp><Y/N>
  ```

  Y : nick name is accepted and reserved
  N : nick name is rejected

  For a query NICK_REQUEST, the 3-byte client-ID will indicate the
  client to which the given nickname is attached. If none is found,
  the ID 000 will be used to indicate failure, and the Y/N flag should
  be set to N.
  For a update NICK_REQUEST, the 3-byte client ID will indicate the
  client to which the nickname is attached (if the update was successful,
  then it will be the ID of the NICK_REQUEST originator). The Y/N flag
  will be set as appropriate.

3.2.2 Group membership messages:

These messages will use the UNRELIABLE DATA outer header type. The
various messages in this set include :

- WHO : Sent by any client to the designated server.

```
  The data part in ASCII text:

  WHO<sp><3 byte client Id in ASCII>

  Where <3 byte client Id in ASCII> is an optional argument, which specifies the client Id
  to be probed. If no client Id is provided, the designated server returns all the user
  currently connected to the E-CHIMP network.

- JOIN_GROUP : Sent by a client to a group leader.
  The data part of the UNRELIABLE DATA message for this type would have
  the following ASCII text :

  JOIN<sp><3 byte client Id in ASCII><sp><1-12 byte group name>


- LEAVE_GROUP : Sent by a client to a group leader.
  The data part in ASCII text :

  LEAVE<sp><3 byte client Id in ASCII><sp><1-12 byte group name>

  Which client is leaving the group is indicated, since it may be sent
  by another client (other than the one that is leaving) when such a
  silent leave is detected.


- HEART_BEAT : Sent by group leader to each member in its group.
  The data part in ASCII text :

  BEAT<sp><3 byte number of member clients in ASCII><sp><3 byte
  client Id for each client in the group><sp>.....


- GROUP_LEADER_ERROR : Sent by a client to another client who had sent
  a JOIN_GROUP or a LEAVE_GROUP message, where the sending client is
  actually not the correct group leader. This informs the destination
  client that there has been an error in its knowledge of the group leader.
  The data part in ASCII text :

  ERROR<sp><JOIN/LEAVE>

- GROUP_LEADER_REQUEST : Client requesting the group leader’s id
  The data part of the message contains :

  GROUPLEADERREQ<sp><group name upto 12 chars>

  An appropriate response will be sent, always.


- GROUP_LEADER_RESPONSE : Response to above.
  The data part of the message contains :

  GROUPLEADERRESP<sp><group name upto 12 chars><sp><3 bytes of Group
  leader Id in ASCII>

  If the group was not found or if there was a change in group leader
  that was made by the Designated Server, then the requesting client
  has been made the group leader.


- GROUPMEMBER_REQUEST : Client asking server for all users (nicknames)
  attached to it that are part of a given group. This is used to handle
  the LIST group type of client query.
  The data part of the message contains :

  GROUPMEMBERREQ<sp><group name upto 12 chars>
  A blank group name (ie. all bytes are \0) means request for all clients
  attached to this server.


- GROUPMEMBER_RESPONSE : Response to above
  The data part of the message contains :
  GROUPMEMBERRESP<sp><3 bytes number of nicks><sp><12 byte nick for each
  nick included><sp>...


- GROUPLIST_REQUEST : Client to Designated Server with the query for
  names of all groups known to the Designated Server.
  The data part of the message contains :

  GROUPLISTREQ


- GROUPLIST_RESPONSE : Response to above
  The data part of the message contains :

  GROUPLISTRESP<sp><3 bytes number of groups><sp><12 byte group names for
  each group included in this list><sp>...
```

3.2.3 Data messages:

Most user messages, e.g. text, images, should be exchanged between
clients as RELIABLE DATA messages. For extensions like audio
and video streams, it is possible to use UNRELIABLE DATA messages.

The data itself will have the following format:
Destination Nick name or Group Name : 12 bytes (the remaining bytes will
be \0 if the name is less than 12 bytes).
Actual data : variable

4. CLIENT TO CLIENT RELIABILITY USING GO BACK N

All data messages between clients must use reliable TCP
connections (as implemented in PA#3).

5. CLIENT COMMANDS AND RESPONSES

An user passes commands to the client using standard input and
receives responses through standard output.
All these commands are delimited by <CRLF>.

5.1 User Commands

The following are possible commands for the client. These
commands will be typed on the command line interface of the client.

```
        5.1.1 NICK

                DESCRIPTION:

                        The NICK command is used to allow a user to join an
                        E-CHIMP Network with an associated, case-insensitive,
                        nickname that will be used to uniquely identify the
                        user.

                        Note:  All other commands are invalid until the user
                        has successfully joined the network using NICK.

                FORMAT:  nick <SP> <nickname> <CRLF>
                        - A nickname is a string of 1-12 characters where
                        each character is either a letter, or a digit.

                        <nickname> ::= <C><C><C><C><C><C><C><C><C><C><C><C>
                        <nickname> ::= <C><C><C><C><C><C><C><C><C><C><C><C>
                        <C> ::= <letter>|<digit>|<nothing>
                        <letter> ::= any letter from a-z or A-Z
                        <digit> ::= 0|1|2|3|4|5|6|7|8|9
                        <nothing> ::= empty value

                CLIENT RESPONSES:

                        -User successfully added to network
                        FORMAT:  10 <Optional Text> <CRLF>

                        -User could not be added to network, because a user
                        already exits in the network with the given nickname
                        FORMAT:  21 <Optional Text> <CRLF>

                        -The provided user name is not in valid format
                        FORMAT:  22 <Optional Text> <CRLF>

                        -The NICK command could not be performed because the
                         user has already joined the network
                        FORMAT:  23 <Optional Text> <CRLF>

                        <Optional Text> ::= <SP> <Text>
                        <Text> ::= any string of typeable characters
                                   excluding <CRLF>


        5.1.2 JOIN

                DESCRIPTION:

                        The JOIN command is used to allow user to join a
                        group on an E-CHIMP Network the user has connected to
                        using the NICK command.

                FORMAT:  join <SP> <groupname> <CRLF>

                        -A groupname is a string of 2-12 characters, where
                        the first character is the pound sign, and the
                        following characters are either a letter or number.

                        <groupname> ::= #<C><C><C><C><C><C><C><C><C><C><C>
                        <C> ::= <letter>|<digit>|<nothing>
                        <letter> ::= any letter from a-z or A-Z
                        <digit> ::= 0|1|2|3|4|5|6|7|8|9
                        <nothing> ::= empty value

                CLIENT RESPONSES:

                        -User successfully joined to the group, <groupname>
                        FORMAT:  11 <SP> <groupname> <Optional Text> <CRLF>

                        -The provided group name, <groupname>, is not in
                        valid format
                        FORMAT:  24 <SP> <groupname> <Optional Text> <CRLF>

                        -The function could not be preformed because the user
                        has not connected to the network
                        FORMAT:  20 <Optional Text> <CRLF>

                        <Optional Text> ::= <SP> <Text>
```

```
                    <Text> ::= any string of typeable characters
                               excluding <CRLF>


        5.1.3 LEAV

                DESCRIPTION:

                        The LEAV command is used to allow a connected user to
                        leave a group that the user has joined using the JOIN
                        command.

                FORMAT: leav <SP> <groupname> <CRLF>

                        -A groupname is a string of 2-12 characters, where the
                        first character is the pound sign, and the following
                        characters are either a letter or number.

                        <groupname> ::= #<C><C><C><C><C><C><C><C><C><C><C>
                        <C> ::= <letter>|<digit>|<nothing>
                        <letter> ::= any letter from a-z or A-Z
                        <digit> ::= 0|1|2|3|4|5|6|7|8|9
                        <nothing> ::= empty value

                CLIENT RESPONSES:

                        -User successfully left the group, <groupname>
                        FORMAT:  12 <SP> <groupname> <Optional Text> <CRLF>

                        -The provided group name, <groupname>, is not in
                        valid format
                        FORMAT:  24 <SP> <groupname> <Optional Text> <CRLF>

                        -The function could not be performed because the user
                        is not connected to the group, <groupname>
                        FORMAT:  25 <SP> <groupname> <Optional Text> <CRLF>

                        -The function could not be preformed because the user
                        has not connected to the network
                        FORMAT:  20 <Optional Text> <CRLF>

                        <Optional Text> ::= <SP> <Text>
                        <Text> ::= any string of typeable characters
                                   excluding <CRLF>


        5.1.4 SEND

                DESCRIPTION:

                        The send command is used to allow a user to send a
                        message to another user or another group in an
                        E-CHIMP Network. The command will attempt to send a
                        message to a user if the specified destination is in
                        valid nickname format, or to a group if the
                        destination is in valid groupname format.

                FORMAT:  send <SP> <Destination> <CRLF> <Message> <Delimiter>

                        <Destination> ::= <nickname>|<groupname>
                        <nickname> ::= <C><C><C><C><C><C><C><C><C><C><C><C>
                        <groupname> ::= #<C><C><C><C><C><C><C><C><C><C><C>
                        <C> ::= <letter>|<digit>|<nothing>
                        <letter> ::= any letter from a-z or A-Z
                        <digit> ::= 0|1|2|3|4|5|6|7|8|9
                        <nothing> ::= empty value

                        <Message> ::= any sequence of typeable characters
                                      that does not include the sequence
                                      <CRLF>.<CRLF>

                        <Delimiter> ::= <CRLF>.<CRLF>

                CLIENT RESPONSES:

                        -The function could not be preformed because the user
                        has not connected to the network
                        FORMAT:  20 <Optional Text> <CRLF>

                        -The action could not be performed because the given
                        destination, <Destination>, was not found.
                        FORMAT:  26 <SP> <Destination> <Optional Text> <CRLF>

                        -The action could not be performed because the given
                        the client is not joined to this group.
                        FORMAT:  27 <SP> <groupname> <Optional Text> <CRLF>

                        <Optional Text> ::= <SP> <Text>
                        <Text> ::= any string of typeable characters
                                   excluding <CRLF>

        5.1.4 SENDFILE

                DESCRIPTION:
                        The sendfile command is used to allow a user to send a
                        mime encoded file to another user or another group in an
                        E-CHIMP Network. MIME encoding is to be implemented
                        manually as a part of the project. The command will attempt to send
a

                        message to a user if the specified destination is in
                        valid nickname format, or to a group if the
                        destination is in valid groupname format.


                FORMAT:  send <SP> <Destination> <CRLF> <File Name> <Delimiter>

                        <Destination> ::= <nickname>|<groupname>
                        <nickname> ::= <C><C><C><C><C><C><C><C><C><C><C><C>
                        <groupname> ::= #<C><C><C><C><C><C><C><C><C><C><C>
                        <C> ::= <letter>|<digit>|<nothing>
                        <letter> ::= any letter from a-z or A-Z
                        <digit> ::= 0|1|2|3|4|5|6|7|8|9
                        <nothing> ::= empty value

                        <File Name> ::= Name of the file that needs to be
                        transferred to another user. The file has to be
                        mime encoded before it is transferred on the
                        network. The destination on receiving, should
                        perform mime decoding to get the actual contents.
                        The file name should not contain the sequence
                        <CRLF>.<CRLF>

                        <Delimiter> ::= <CRLF>.<CRLF>

                CLIENT RESPONSES:

                        -The function could not be preformed because the
                        file <File Name> does not exist.
                        FORMAT:  20 <Optional Text> <CRLF>
```

```
                        -The function could not be preformed because the user
                        has not connected to the network
                        FORMAT:  20 <Optional Text> <CRLF>

                        -The action could not be performed because the given
                        destination, <Destination>, was not found.
                        FORMAT:  26 <SP> <Destination> <Optional Text> <CRLF>

                        -The action could not be performed because the given
                        the client is not joined to this group.
                        FORMAT:  27 <SP> <groupname> <Optional Text> <CRLF>

                        <Optional Text> ::= <SP> <Text>
                        <Text> ::= any string of typeable characters
                                   excluding <CRLF>


        5.1.5 LIST

                DESCRIPTION:

                        The LIST command is used to allow a user to query
                        for a list of the current groups in the E-CHIMP
                        Network, or a list of users active in a given group.

                FORMAT:  list <ListQueryType> <CRLF>

                        -The list query type specifies a groupname to query
                        for a list of users in the given group, or nothing
                        to query for a list of current groups.

                        <ListQueryType> ::= <SP><groupname>|<nothing>
                        <groupname> ::= #<C><C><C><C><C><C><C><C><C><C><C>
                        <C> ::= <letter>|<digit>|<nothing>
                        <letter> ::= any letter from a-z or A-Z
                        <digit> ::= 0|1|2|3|4|5|6|7|8|9
                        <nothing> ::= empty value


                CLIENT RESPONSES:

                        -Successful response with current list of groups
                        FORMAT:  13 <CRLF> <Group List> <CRLF>

                        -Successful response with current list of users
                        in given group <groupname>
                        FORMAT:  14 <CRLF> <groupname> <User List> <CRLF>

                        -The action could not be performed because the given
                        group, <groupname>, was not found.
                        FORMAT:  26 <SP> <groupname> <Optional Text> <CRLF>

                        -The function could not be performed because the user
                        has not connected to the network
                        FORMAT:  20 <Optional Text> <CRLF>

                        <Group List> ::= <SP><groupname><GroupList>|<nothing>
                        <User List> ::= <SP><nickname><User List>|<nothing>
                        <nickname> ::= <C><C><C><C><C><C><C><C><C><C><C><C>
                        <Optional Text> ::= <SP> <Text>
                        <Text> ::= any string of typeable characters
                                   excluding <CRLF>

        5.1.6 WHO
```

```
                DESCRIPTION:

                        The who command is used to allow a user to query for
                        a list of users currently joined on an E-CHIMP
                        Network, or if a specified user exists.

                FORMAT: who <WhoQueryType> <CRLF>

                        -The who query type specifies a nickname, to query if
                        a user with the given nickname exists, or nothing to
                        query for a list of all users on the network.

                        <WhoQueryType> ::= <SP><nickname>|<nothing>
                        <nickname> ::= <C><C><C><C><C><C><C><C><C><C><C><C>
                        <C> ::= <letter>|<digit>|<nothing>
                        <letter> ::= any letter from a-z or A-Z
                        <digit> ::= 0|1|2|3|4|5|6|7|8|9
                        <nothing> ::= empty value

                CLIENT RESPONSES:

                        -Successful response containing list of active users
                        FORMAT:  15 <CRLF> <User List> <CRLF>

                        -The specified user, <nickname>, exists.
                        FORMAT:  16 <CRLF> <username> <Optional Text> <CRLF>

                        -The specified user, <nickname>, does not exist.
                        FORMAT:  26 <CRLF> <nickname> <Optional Text> <CRLF>

                        -The function could not be preformed because the user
                        has not connected to the network (i.e. does not have
                        a nick name registered at the server).
                        FORMAT:  20 <Optional Text> <CRLF>

                        <User List> ::= <SP><username><User List>|<nothing>
                        <username> ::= <nickname><SP><clientID><sp><IPaddr><sp><porrt>
                        <nickname> ::= <C><C><C><C><C><C><C><C><C><C><C><C>
                        <clientID> ::= <digit><digit><digit>
                        <Optional Text> ::= <SP> <Text>
                        <Text> ::= any string of typeable characters
                                   excluding <CRLF>




5.2 CLIENT RESPONSES

        The following are a list of possible responses a user can receive
        from a client.  All responses are preceded by a number ranging
        from 10-16 for positive acknowledgments or 20-28 for
        negative acknowledgments.
        Type 30 indicates user data received from a remote client.

        -User successfully added to network
        FORMAT:  10 <Optional Text> <CRLF>

        -User successfully joined to the group, <groupname>.
        FORMAT:  11 <SP> <groupname> <Optional Text> <CRLF>

        -User successfully left the group, <groupname>.
        FORMAT:  12 <SP> <groupname> <Optional Text> <CRLF>

        -Response contains current list of groups in the network.
        FORMAT:  13 <CRLF> <Group List> <CRLF>
        -Response contains current list of users in given group <groupname>
        FORMAT:  14 <CRLF> <groupname> <User List> <CRLF>

        -Response contains list of active users in the network.
        FORMAT:  15 <CRLF> <User List> <CLRF>

        -The specified user, <nickname>, exists in the network.
        FORMAT:  16 <SP> <nickname> <Optional Text> <CRLF>

        -The function could not be preformed because the user has not
        connected to the network
        FORMAT:  20 <Optional Text> <CRLF>

        -User could not be added to network, because a user already exits in
        the network with the given nickname
        FORMAT:  21 <Optional Text> <CRLF>

        -The provided user name is not in valid format
        FORMAT:  22 <Optional Text> <CRLF>

        -The NICK command could not be performed because the
        user has already joined the network
        FORMAT:  23 <Optional Text> <CRLF>

        -The provided group name, <groupname>, is not in valid format
        FORMAT:  24 <SP> <groupname> <Optional Text> <CRLF>

        -The function could not be performed because the user is not connected
        to the group, <groupname>
        FORMAT:  25 <SP> <groupname> <Optional Text> <CRLF>

        -The action could not be performed because the given destination,
        <Destination>, was not found.
        FORMAT:  26 <SP> <Destination> <Optional Text> <CRLF>

        -Invalid command
        FORMAT:  28 <Optional Text> <CRLF>

        - User message received by the client
        FORMAT: 30 <SP> <NICKNAME OF SENDER> [<SP> <GROUP NAME>] <CRLF>
                <Text>

BNF FOR CLIENT RESPONSES:

<Destination> ::= <nickname>|<groupname>
<nickname> ::= <C><C><C><C><C><C><C><C><C><C><C><C>
<groupname> ::= #<C><C><C><C><C><C><C><C><C><C><C>
<C> ::= <letter>|<digit>|<nothing>
<letter> ::= any letter from a-z or A-Z
<digit> ::= 0|1|2|3|4|5|6|7|8|9
<Group List> ::= <SP><groupname><GroupList>|<nothing>
<User List> ::= <SP><nickname><User List>|<nothing>
<nothing> ::= empty value
<Optional Text> ::= <SP> <Text>
<Text> ::= any string of typeable characters excluding <CRLF>
```

7. CONCLUSIONS

Questions and comments regarding the e-CHIMP protocol should be
posted to the mailing list, cs640-1list@cs.wisc.edu

Additional Notes:
=================

```
Here is a simplified version of the description on what is
expected and what will be tested on the clients.
Each server and client will be assigned an unique ID for our
convenience. This is the same ID as used in the Outer Header
of the packets.
There is a unique mapping between a <hostname,port> pair and
the client/server ID.


Run the server as follows:
./server -p <port> -i <id>
where <port> is the port number in which the server runs,
id is the ID for the server.

Run the client as follows:
./client -p <port> -s <server-host> -q <server-port> -i <id>
<port> is the port number in which the client runs,
<server-host> is the hostname in which the server operates,
<server-port> is the server’s port number,
<id> is the ID for the client.


The client should present a command line interface as follows:
Lines beginning with # are comments.
Lines beginning with > is what the user types in stdin.
Lines beginning with : is what the client outputs to stdout.


## Attempt to list other clients when not joined to the server.
## 20 in the response is the return code.
> list
: 20 Not joined to server

## Request a nick name (message will get sent to server)
## 10 in the response is the return code (for successful operation).
> nick nick100
: 10 nick nick100 added

## List all groups and appropriate response
## is obtained from the server. 13 is return code.
> list
: 13
: group1
: group2

## List all nick names, and appropriate response obtained
## from server. You will also display the IDs, IP-addr, and ports.
## 15 is the return code.
> who
: 15
: nick1 ID1 IP1 port1
: nick2 ID2 IP2 port2
: nick3 ID3 IP3 port3
: nick100 myID myIP myport

## Send message to a specific nick name.
> send nick2
> beginning of message
> end of message
> .
:

## Set a new nickname (that automatically cancels the
## previous nickname that was registered).
> nick nick200
```

```
: 10 nickname successfully changed to nick200

## We can verify the current list of nicknames at
## the server. 15 is the code for successful return.
> who
: 15
: nick1 ID1 IP1 port1
: nick2 ID2 IP2 port2
: nick3 ID3 IP3 port3
: nick200 myID myIP myport

## Attempt to use a nickname someone is already using.
## Obviously not successful. Return code is 21.
> nick nick1
: 21 nickname nick1 not available

## Check membership of a group.
## 14 is the return code.
> list group1
: 14
: nick2 ID2
: nick3 ID3

## Request to join a group.
> join group1
: 11 joined group group1 successfully

## List group membership to see current members.
> list group1
: 14
: nick2 ID2
: nick3 ID3
: nick200 myID

## Send message to a group. Each member of the group
## should receive a message.
## A message is terminated by a single . on the last line.
> send group1
> beginning of message
> end of message
> .
:

## Send file to a specific nick name.
> sendfile nick2
> File Name
> .
:

## Not allowed to send data to a group I am not joined
## in. Note that I am allowed to list members of a group
## I am not joined to.
> send group2
: 27 not joined to group group2

## A message received from nick2.
## 30 in the message is the status code.
: 30 nick2
: Hello nick100, how are you
: .
```