Conference Communication Channel Protocol

GENERAL

- Implemented as a library
- User space: Spawns child process
- Child implements asynch receive buffering
- Child implements reliability (timeout/retrans/ack)
- DIdn't use threads or SIGIO for portability reasons.

TUPLES

- Conference ID - multicast address
- Site - Hostname. needed by receivers to recognise sender
- Application type - type of applications message should reach. Special types like 'Floor' and 'Announcement' for reaching all applications being provided.
- Application instantiation - Unique ID to distinguish applications of same type on host. (PID)

```
#define MAXTYPE 32
#define MAXADDRESS 64

struct tuple {
        u_long conf_id; /* Note Not IPng Compatible */
        int inst;       /* Unix Specific */
        char *type;     /* Aribitrary really */
        char *address; /* Should be DNS Name Length Compatible */
}
```

MESSAGES

cccp messages will have the following structure:

```
messageType
Destination SET OF {
CHOICE  {
      NameTuple,
      NameID
      }
}
Source CHOICE {
      NameTuple,
      NameID
}
ApplicationSpecificInfo
```

```
tuple   *
cccp_register(u_long conf_id,
           char *app_type);

int
cccp_leave(u_long conf_id);

int
cccp_bind(char *app_type);

int
cccp_unbind(char *app_type);

int
cccp_send(tuple *tolist, int tocount,
unsigned char *message, int msgsize,
int reliability, struct timeval *timeout);
```

```
int
cccp_send_block(tuple *tolist, int tocount,
unsigned char *message, int msgsize,
int reliability, struct timeval *timeout);

int
cccp_receive(tuple *from,
unsigned char *msgbuf, int buflen);

void
cccp_receive_async(tuple *from,
unsigned char *msgbuf, int buflen,
void (*cb)(tuple *from, char *msg, int len));

char    *cccp_error();
```

SESSION MESSAGES

unifying Session Messages from different Media:

Vat, nv—vic—ivs, wb, nt independently generate session messages,

Vat already uses part of the solution for a general session tool:

multicast at ttl=0

CCCP Session Conforming applications use this to co-located session message process

an CCCP Unreliable message of type Session.Announce.

periodicity should be controlled by Van Jacobson's sceme or UCL/INRIA congestion backoff scheme.

FLOOR CONTROL

Entities:

- floor.chair
- floor.holder

Messages:

- haveFloor
- requestFloor
- grantFloor
- amChair

Policies can be none—baton—chaired

FLOOR AND CCCP

haveFloor is sent to floor.*

requestFloor is sent to

1. holder, if baton and known
2. floor.chair, floor.holder, if known
3. floor.* otherwise

grantFloor goes from holder to chair if chaired

If baton, and someone leaves conf, grantFloor goes to floor.*

amChair are periodic, and go to floor.*

chair keep queue of requests, allocate the floor in FIFO order

protocol needed to solve lost baton

to interact with other events, fm binds to MEDIA e.g. to see MEDIASTART from audio programs

All kinds of other Policies Envisaged!