Asian Journal of Information Technology 6 (7): 778-784, 2007
ISSN: 1682-3915

An Extension of the IETF RFC 4376 Based on Distributed Floor Control Servers

Abdelali Ibriz
Department of Computer Sciences, Sidi Mohamed Ben Abdellah University,
ST Fès B.P. 2427, Route d'Immouzer Fès, Maroc

Abstract: Floor control is a means to manage joint or exclusive access to shared resources in a synchronous groupware. Floor control in CSCW (Computer Supported Cooperative Work) is a metaphor for "assigning the floor to a speaker", which is applicable to any kind of sharable resources within conferencing and cooperation environments. This study discusses an extension of a floor control protocol for internet synchronous groupware. We purpose how to extend RFC4376. Extension concerns two levels: Distribution and the organization of Floor control servers, for the same cooperative session and separation betweenfloor control policy and floor control protocol. Floor control servers mediate the access to shared remote resources in a synchronous cooperative session. Floor control servers, for the same cooperative session, are distributed and organized in a logical tree. Also they are used for routing floor control messages and for storing the floor participant requests.

Key words: Internet video conferencing, CSCW, floor control and policy, SIP, RFC4376

INTRODUCTION

Conference applications often have shared resources such as the right to talk, input access to a limited-bandwidth video channel, or a pointer or input focus in a shared application (Ellis *et al*., 1991).

In many cases, it is desirable to be able to control who can provide input (send/write/control, depending on the application) to the shared resource.

Floor control enables applications or users to gain safe and mutually exclusive or non-exclusive input access to the shared object or resource. The floor is an individual temporary access or manipulation permission for a specific shared resource (or group of resources) (Ellis *et al*., 1991).

Floor control may be used together with the Conference Policy Control Protocol (CPCP) (Floyd *et al*., 1995) or it may be used as an independent stand-alone protocol, e.g., with SIP but without CPCP.

With the IP multicasting more powerful cooperation multimedia applications has been gradually developed (Clark, 1992; Floyd *et al*., 1995; Handly *et al*., 1997). However, the participants' number of such application will scale to hundreds/thousands or more. Such application will often deploy floor control techniques (Dommel *et al*., 1999a,b; Dommel *et al*., 2000).

At present there are several standards and non-standard model for controlling the floor in teleconferencing environment. IETF gives the simple conference control protocol (Handley *et al*., 1999). The RFC 4376 defines the requirements for a floor control protocol for multiparty (Koskelainen *et al*., 2004). IUT presents H.332 by only dividing the conference participants into two parts (presidium and audience) (IUT, 1998). This implies an ambiguous implementation between the floor control policy and its operations.

Existing non-standard model for floor control will run correctly on top of routing protocol. These protocols are designed to address a known centralised floor controller. A first object-oriented architecture for teleconferencing with floor control was proposed by Aguilar *et al*. (1986) where changes are made on shared parts by asynchronous broadcasts messages without using multicast protocol.

We remarks that is difficult to construct a large-scale floor control for distributed collaborative environment without distributing the service informations, the separation between the services and its application (policy) and using scalable transport protocols.

The contributions of the floor control protocol proposed in this study are:

- Servers, providing Floor control services, are distributed and organized in a logical tree correlated to an underlying multicast IP schema.
- A separation between the floor control protocol and the policy used to assign the floor.

The first contribution supposes that inside a cooperative application there can be many sessions. Each session contains a connected group of nodes. In addition, each session has a floor controller server. These servers are connected by the reliable multicast tree. While the second considers that each session has its own policy. A floor policy algorithm is embedded in the session server and interacts with the floor server. This later carried out an algorithm to reach the floor and the floor holder by using the IETF Session Initiation Protocol (SIP) services.

THE IETF RFC 4376

This study briefly outlines the IETF RFC 4376 floor control.

Model: The model for floor control is composed of three logical entities: A single floor control server, one or more floor chairs (moderators) and any number of regular conference participants.

A floor control protocol is used to convey the floor control messages among the floor chairs (moderators) of the conference, the floor control server and the participants of the conference. A centralized architecture is assumed in which all messages go via one point, the floor control server. Processing (granting or rejecting) floor control requests is done by the one or more floor chairs or by the server itself, depending on the policy.

Floor requests from the participants are received by the floor control server and kept (at the level of the floor control protocol) in a floor request set (i.e., are not ordered in any particular fashion). The current floor holders are reflected in a current floor holder set. Floor chairs are capable of manipulating both sets to grant, revoke, reject and pass the floor (for example). The order in which requests are processed, whether they are granted or rejected and how many participants obtain a floor simultaneously are determined by a higher-layer application operating on these sets and are not confined by the floor control protocol.

A floor is associated with one or more media sessions. The centralized conference server manages the floors and thus controls access to the media sessions. There are 2 aspects to this:

- The server maintains and distributes consistent state information about who has a certain floor at a certain point in time and does so following some rule set. This provides all participants with the necessary information about who is allowed to speak (for example), but relies on a cooperative behavior among all participants.
- In addition, to prevent individuals from ignoring the "hints" given by the floor control server, the latter may (e.g., in cooperation with other functional entities) enforce compliance with floor status, e.g., by blocking media streams from participants not entitled to speak. The floor control server controls the floors at least at the signalingsecond considers that each session has its own policy. A floor policy algorithm is embedded in the session server and interacts with the floor server. This later carried out an algorithm to reach the floor and the floor holder by using the IETF Session Initiation Protocol (SIP) services.

Integration with conferencing: Floor control itself does not support privileges such as creating floors and appointing floor chairs and handing over chair privileges to other users (or taking them away). Instead, some external mechanism, such as conference management (e.g., CPCP or web interface for policy manipulation) is used for that.

The conference policy (and thus the conference owner or creator) defines whether floor control is in use or not.

Typically, the conference owner creates the floor (s) using the conference policy control protocol (or some other mechanism) and appoints the floor chair. The conference owner can remove the floor anytime

The floor chair just controls the access to the floor(s), according to the conference policy.

A floor control server is a separate logical entity, typically co-located with focus and/or conference policy server. Therefore, the floor control server can interact with the focus and conference policy server and media servers as needed.

Assumptions about a conference policy: The floor control protocol is supposed to be used to manage access to shared resources in the context of a conference. It is up to this conference to define the rules for the operation of the floor control protocol (Bormann *et al*., 2001).

Furthermore, a conference policy control protocol may define mechanisms that alter those rules during the course of a conference (Bormann *et al*., 2001; Koskelainen *et al*., 2004).

The conference policy is expected to define the rules for floor control, which implies in particular that it is not the responsibility of the floor control protocol to establish or communicate those rules.

In general, it is assumed that the conference policy also defines who is allowed to create, change and remove a floor in a conference. Conference participants and floor chairs should be able to get and set floor-related parameters (IFTF RFC 476).

EXTENSION FRAMEWORK

System architecture consideration level: Figure 1 is an example of our system architecture that shows the interactions between two distributed teleconferencing entities and the various protocols in use.

This architecture consists in 3 layers: The cooperative application layer, the cooperative session layer and the videoconferencing system layer. Detailed description of this architecture is given in Ibriz and Erradi (2004).

Users access different types of visible services via an Application Programming Interfaces (API). The mains services are provided by the session layer:

- Cooperation management realized by Session manager that maintains the multicast addresses, the cooperation agent addresses and port numbers used by the cooperative session.
- Cooperation session control carried out by the Cooperation Sessions servers. It provides control services such as inviting, joining and leaving a conference (Ibriz *et al*., 1999; Bormann *et al*., 2001). In addition, the cooperative session process maintains the appropriate links with the invoked media. Among the actions performed by the cooperation session we found: Initiate, pause, resume and stop.
- Floor Control Services solves the session resources conflicting problem between different participants and help coordinate and synchronize them in the collaboration work. All resources are represented in logical structures called Shared Space. A shared space process is created to manage critical session for shared resources. This process maintains this data in a consistent state during the cooperation session.

Several cooperative applications may be instantiated in parallel. Each instance defines a participant node. Specific application may be a course, a directed work, an exam, a telepointer, or any other multimedia application. Once a cooperative application is invoked by the end user, the cooperation manager sends a request to the cooperation session level to create a cooperation session process or to join an existing session.

Floor control protocol consideration level: Shared resources in networked cooperative applications are not based on direct user-resource transaction, but rather on user-session-user interactions.

Every session server corresponds to a nodes group. The sites of every group are connected to the same session server. Session server can be also connected to other session server. All nodes are connected by the reliable multicast tree. Figure 2 shows the root floor server which may send a floor control messages to other receiving servers in the tree. The receiving servers can be session servers and/or floor servers.

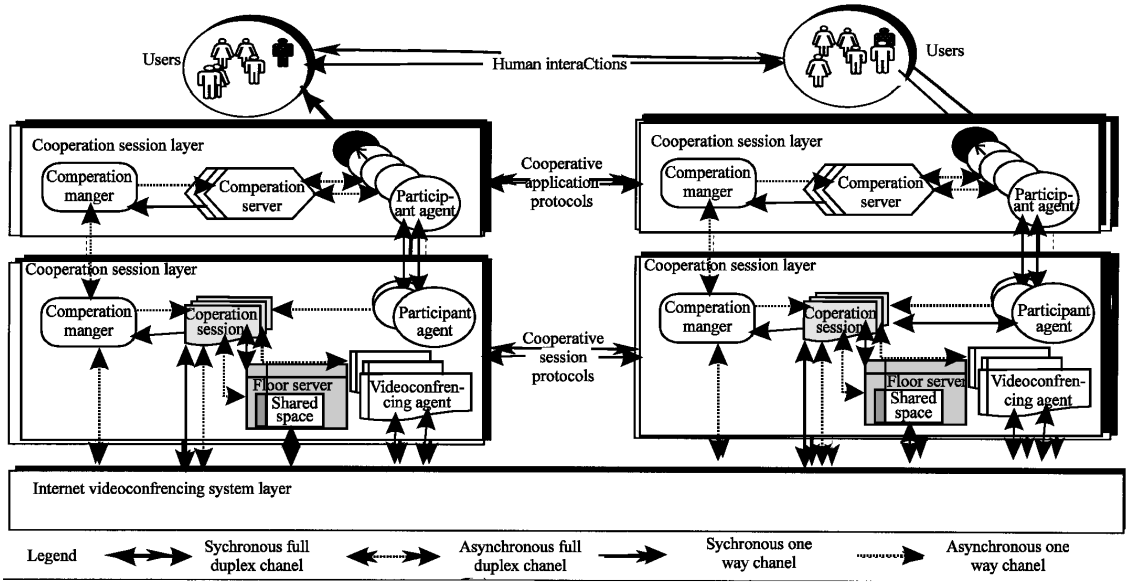

Fig. 1: Layered Cooperative architecture

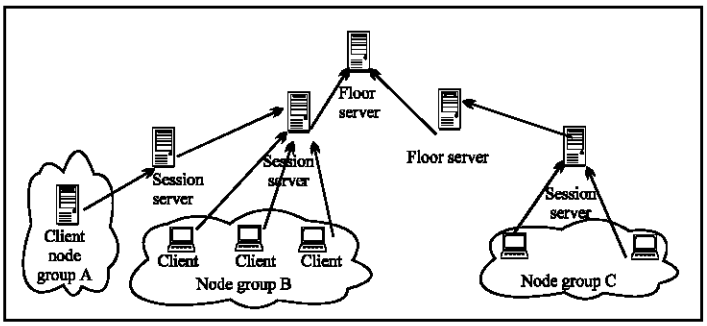

Fig. 2: Tree organization of servers

To access the floor server, the user of a floor service should first send a request to the session server. The later forwards the received request to the floor server connected to such session. Each floor server manages a set of resources and grants access to such resources.

The decision of assigning a floor is made in general on the basis of a given policy. For instance one can use a token permission to access the floor (Housni and Trehel, 2001). In fact, the proposed architecture delegates the choice of appropriate policy to the session server. Therefore, in a cooperative work we can find several sessions using different policies. For the resources localization, associated to a given floor, we adopt the SIP localization services.

The floor server used as an API for routing floor control primitives and storing the sites requests. For instance, the "release_floor" primitive involves the routing of an "expand_floor" primitive to the floor servers.

Floor service primitives exchanged between the different entities of the floor control protocol are categorized as follows:

Between client and session server:

- "Floor_Access": The user intention to access a floor
- "Floor_Wait": This request is sent to the floor server but no reply is transmitted.
- "Floor_ denied": The user not authorized to use the requested floor.
- "Floor_grant" : the floor is attributed to the user.

Between session server and SIP server:

- "Register": When a floor is created it must be registred as a URI (Inform Resource Information) within the SIP server.
- "Response 200" : SIP acknowledgement for registration operation.

Between session server and floor server:

- "create_floor": To create a new floor,
- "check_floor": Return the Ids of the floor owner,
- "request_floor": To request floor for a subscribed participant,
- "grant_floor": Attributes the floor to a session server.
- "lock_floor": Locks shared resources for an exclusive use.
- "revoke_floor": Is used by the floor owner to revoke floor helds by others before leaving a session.
- "kill_floor": to kill a floor,
- "release_floor": Frees a floor.

Between floor server and floor server: In addition to the exchanged primitives between session server and floor server presented above we find the primitive:

- "Expand_floor": Sends the current state floor table to all floor servers when modification is occurred.

Between floor server and SIP server:

- "INVITE": Is issued from Floor server to localize the Resource and to read its state. Resources are addressed as URI's (Uniform Resources Information). SIP define its own URI, but its header fields can carry other URI's, such as http page, mailto, phone or other specific resource such as media.
- "Resp 302": The SIP resources localization response.

A typical utilization of these primitives is depicted in Fig. 3. A client initiates a "floor_access" request to be

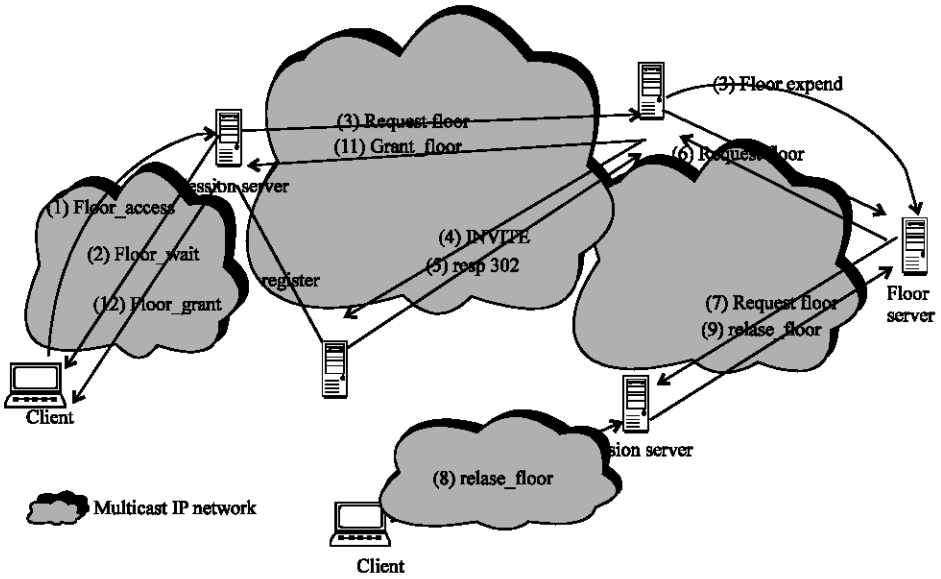

Fig. 3: An example of floor primitives' exchange

sent to its session server (1). A waiting response is replied to the client (2) and the request is forwarded to the local floor server as a "floor_request" message (3). This floor must be registered beforehand at the SIP server as an URI (Register). The floor server looks up the SIP server for localizing the Floor server handling the requested floor (4,5) . The floor server then proxies the request to the SIP localized floor server (6) and store it . This former knows about the floor holder and its session. It then asks the session server to release this floor according to the floor policy algorithm result (7,8).

If the floor policy algorithm result leads to "release_floor", the session server replies with a "release_floor" response to the attached floor server (9). It then proxies this response to the requester floor server (10) as a "grant_floor" message and so on until reaching the session server that initiated the request (11).

Therefore, the floor is granted to the client requesting the floor (12). This scenario ends by updating the floor state (13) by the requester floor server and become the holder floor server.

In the following, we give the description of a floor assignment policy algorithm based on the floor control protocol presented above.

Floor control policy: Now, we have presented a floor control protocol upon a tree organization of servers, we describe in this section our floor assignment policy algorithm wich uses a token based permission approach. An abstract view of the floor assignment policy consists in the following main procedures (Fig. 4): "Init" procedure "Handle_floor" procedure, "Release_floor" procedure, "Request_floor" procedure and "Route_control_ message" procedure. The code of these procedures incarnates the used floor assignment policy technique.

The "Init" procedure starts the first floor server as root, sets the floor to free and initializes the token. A floor access from a simple client invokes a "Request_floor " procedure at the session server. The "Handle_Floor" procedure grants the floor to the requester client. The "Release_floor" procedure frees the floor from a given client. The Route_Control _Message procedure routes the control messages among nodes of the logical tree.

We give a detailed view of the above algorithm trough a specific configuration example. Figure 5 a and b presents 2 configurations of the logical tree. The nodes of the tree represent the floor servers. The FH node designates the floor server of the client holding the floor. The FS nodes designate the simples floor servers. Every floor server owns a local variable called its "father" which indicates the direction of the root and manages a queue for storing floor requests.

- At the initial configuration, FS4 requests the floor to its father and considers itself to be the new root. Thus, a series of request messages travel along the nodes of the tree. At each passage, the requester

```
1. Procedure init
begin
set S =Cs; /* Multicast address and port */
set floor; / * Start the floor server as tree root*/
floor.state:=idle;
nexti:=lasti:=1; /* the first client*/
If i=1 then token_state:=init_token_state; endif
end;

2. Procedure Request_floor ( participant x, floor);
begin
Case floor.state of
Idle: if nexti=lasti=x the token_state:=nexti
        Handle_floor(x); endif
Busy: Route_control_message(request_floor, i, S, token_state));
Requested:Route_control_message(shrink_floor, i, S, token_state);
Deny :Route_control_message(kill_floor, i, S, token_state);
Endcase
Route_control_message(expand_floor, i, S, token_state);
End;

3. Procedure Handle_floor(participant x);
begin
if token_state= x then
S.root:= x;
reconfigure(S, x);
floor.state=busy;
x.floor_state:=holder;
Route_control_message(expand_floor, x, S, token_state);
else
x.floor_state:=waiting;
Nexti=x;
Endif

4. Procedure release_floor ( participant.x, floor);
if token_state:=x then
 floor.state:=idle
token_state:=next_token;
Route_control_message(release_floor, i, S, tohen_state);
Route_control_message(expand_floor, i, S, token state);
End;

5. Procedure Route_control_message(message M, Client x, CS S,
token token_state)
begin
If x.floor_state<>holder then
Foreach x.child do Route_control_message( M, child, S, token_state)
Else if x._floor_state=nil then
S.Root := next x
Route_control_message( M, Root, S, token_state)
endif
Endif
End
```

Fig. 4: Main algorithm procedures

floor server is queued (for instance FS4 in the FS2 queue and FS2 in the FH queue). After FH releases the floor then the token must be transmitted to the FS4 according to the token circulation (represented by dashed lines).

- Thus FH reply the grant message to FS2 and updates its queue. This is done until FS4. The tree is reorganized and the new root is FS4.

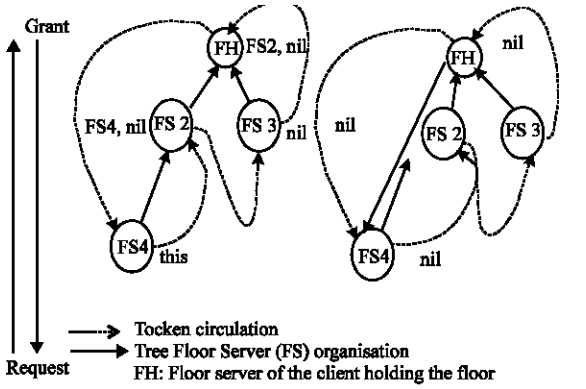

Fig. 5: A configuration example of the floor policy algorithm

CONCLUSION

In this study, we have presented a An Extension of the IETF RFC 4376 based on Distributed Floor Control Servers. Based on our layered cooperative architecture, floor servers are defined as a part of the session layer services. The floor servers are distributed and organized in a logical tree correlated to an underlying multicast IP schema.

The floor assignment policy does not depend on the floor protocol services. This approach allows the implementation of different floor policies using the same floor control protocol. Moreover, within the same cooperative session, nodes group can use different floor policies. Scalability of this protocol is ensured by using the multicasting transport services.

The protocol and its procedures was specified and verified under Promela/Spin. We have implemented this protocol with a kernel functionality schema and examples of policies using Java/RMI language. The proposal for this work as an IETF RFC constitutes our further works.

REFERENCES

Aguilar, L. *et al*., 1986. Architecture for a multimedia teleconferencing system. In Proc. ACM SIGCOMM., pp: 126-136.

Bormann, C. *et al*., 2001. Simple Conference Control Protocol. Internet Draft, draft-ietf-mmusic-sccp-01.txt.

Clark, W.J., 1992. Multimedia Conferencing, IEEE. Commun. Mag., pp: 44-50.

Dommel, H.P. *et al*., 1999a. Group coordination support for synchronous Internet collaboration. IEEE Internet Computing Magazine, Special Issue on Multimedia and Collaborative Computing over the Internet, pp: 74- 80.

Dommel, H.P. *et al*., 1999b. Efficacy of Floor Control Protocols in Distributed Multimedia Collaboration, Cluster Computing, 2: 17-33.

Dommel, H.P. *et al*., 2000. Ordered End-to-End Multicast for Distributed Multimedia Systems. 33rd Annual Hawaii International Conference on System Sciences (HICSS-33), Maui, Hawaii.

Ellis *et al*., 1991. Groupware: Some Issues and Experiences Communications, ACM., 34: 39-58.

Floyd *et al*., 1995. A Reliable Multicast for Light-Weight Sessions and Application for Level Framing, ACM SIGCOMM., pp: 342-356.

Handley, M. *et al*., 1999. SIP: Session Initiation Protocol (SIP), IETF Draft.

Handley, M. *et al*., 1997. The internet multimedia conferencing architecture', Internet Draft, Internet Engineering Task Force, Work in progress.

Housni, A. and M. Tréhel, 2001. A New Distributed Mutual Exclusion Algorithm for two Groups, ACM Symposium on Applied Computing (SAC), Track on Parallel and Distributed Processing, Las Vegas, USA., pp: 531-538.

Ibriz, A., M. Erradi , 2004. Cooperative Session Control Over SIP, proceeding of MSCEAI, Sousse Tunisia, pp: 223-234.

Ibriz, A. *et al*., 1999. A Layered Cooperative Architecture and its Application in Tele-teaching Framework, in proceeding of PDPTA, Las Vegas, USA., 3: 1516-1522.

IUT, 1998. H.323 extended for loosely coupled conferences.

IUT, 1998. Paket Based multimedia communication systems.

Koskelainen, P. *et al*., 2004. The Conference Policy Control Protocol (CPCP), Work in Progress, Nokia 102 Corporate Park Drive, White Plains, NY 10604, USA.