

# UMPIRE: A Universal Moderator for the Participation in IETF Remote Events

Simon Pietro Romano

## ABSTRACT

UMPIRE provides seamless meeting interaction among remote and local participants. It uses the BFCP, an IETF standard for moderation. BFCP introduces automated floor control functions to a centralized conferencing environment. This article discusses the design and implementation of the UMPIRE system and highlights the most notable solutions we devised to handle varied requirements and constraints. We also discuss the lessons learned while experiencing in the first person how the application of research results that have eventually led to new standards still must confront a number of minor yet concrete issues that might completely undermine the overall process of wide adoption by the community.

## BACKGROUND, RATIONALE AND MOTIVATION

UMPIRE provides seamless meeting interaction among remote and local participants. It uses the Binary Floor Control Protocol (BFCP), a standard for moderation. BFCP introduces automated functions for a centralized conferencing environment. The project has been motivated by the ongoing efforts within the IETF (Internet Engineering Task Force) to standardize mechanisms for enabling remote participation at meetings. At that time, work was in full swing and was being formalized in a (now expired) Internet draft<sup>1</sup> specifically devoted to this task. The draft in question was actually being produced at the request of the IETF Administrative Oversight Committee (IAOC), which issued an ad hoc request for proposals at the end of October 2011.<sup>2</sup> The draft contains discussion (section 2.3.4.1) dedicated to the task of moderation. Titled “*Floor Control for Chairs for Audio from Remote Attendees*,” it contains a list of requirements, including:

**\*\*Requirement 08-31\*\*:** Remote attendees **MUST** have an easy and standardized way of requesting the attention of the chair when the remote attendee wants to speak. The remote attendee **MUST** also be able to easily cancel an attention request.

**\*\*Requirement 08-33\*\*:** The floor control portion of the Remote Participation Sys-

tem **MUST** give a remote attendee who is allowed to speak a clear signal when they should and should not speak.

**\*\*Requirement 08-34\*\*:** The chair **MUST** be able to see all requests from remote attendees to speak at any time during the entire meeting (not just during presentations) in the floor control system.

**\*\*Requirement 08-35\*\*:** The floor control system **MUST** allow a chair to easily mute all remote attendees.

**\*\*Requirement 08-36\*\*:** The floor control system **MUST** allow a chair to easily allow all remote attendees to speak without requesting permission; that is, the chair **SHOULD** be able to easily turn on all remote attendees mics at once.

UMPIRE is currently capable of meeting most of the requirements in that list.

We also took inspiration from related mailing list discussions. To get an idea of such discussions, consider the following thread on the IETF’s Working Group Chairs mailing list: “*Getting Taipei remote participants’ input*.” It includes a message from the author of this article,<sup>3</sup> who was proposing to use RFID (Radio Frequency Identification) tags to trigger BFCP requests by in-person participants, in much the same way as the floor requests generated by remote participants. In answer to such message, Dave Crocker wisely stated the following:<sup>4</sup>

Suggestions like the above sound appealing. Unfortunately, they are far beyond current products and making them useful is considerably more difficult than the suggestions imply. That doesn’t mean they should be ignored, but we need to be careful about slipping into the assumption that merely citing a bit of technology means that an issue is resolved. We had an experiment with RFIDs. It was awkward, at best. In the case of queue management, we have at least entering the queue, position in the queue, and the chair’s control of the queue.

A further fundamental contribution to the discussion was also provided by Brian Rosen, who asserted:<sup>5</sup>

People worry about RFID, but I like it because it’s a faster read. All I think you want is a reader and a visible queue. The

The author is with University of Napoli Federico II.

<sup>1</sup> “Requirements for Remote Participation Services for the IETF,” draft-ietf-genarea-rps-reqs-08.

<sup>2</sup> <http://iaoc.ietf.org/documents/RPS-Specifications-RFP-2011-10-19.pdf>

<sup>3</sup> <http://www.ietf.org/mail-archive/web/wgchairs/current/msg10274.html>

<sup>4</sup> <http://www.ietf.org/mail-archive/web/wgchairs/current/msg10277.html>

<sup>5</sup> <http://www.ietf.org/mail-archive/web/wgchairs/current/msg10280.html>

queue just tells you that the reader read correctly and has you in the queue. The chair gets to change the queue, but that ought to be rare and probably just pick the next person in queue. Remote participants simply imitate the reader action.

The tool could also provide session chairs with the ability to grant “business class” requests (i.e. in the case of cut and thrust debates, or in the presence of intervention of an area director) so that individuals obtain higher priority, essentially putting such requests on top of the queue. This highlights the possibility of applying different activity templates, or paradigms, for common handling of remote and in-person participants, according to different group process modes.

We took into account these considerations and applied the usual IETF approach of having a running code prototype to identify and clarify possible issues and foster discussion. The system we developed provides a proof of concept for a moderation framework built on top of the Meetecho conferencing system [1]. Meetecho is a standards-based conferencing architecture used at IETF meetings for remote participation.

The remainder of this article is organized as follows. The next section helps the reader position this work in the context of ongoing IETF standardization efforts for multimedia conferencing. The third section provides some information about related work in the field of conference moderation. Following that we describe the overall architectural design of the UMPIRE system, whose implementation is briefly sketched in the next section. We then discuss the main issues we faced during the design and implementation phases and provide insight into the lessons learned. Finally, the conclusion summarizes the discussion stimulated by UMPIRE within the IETF community, illustrating the useful feedback we gathered and the envisioned direction of our future efforts.

## CONTEXT

The IETF has devoted much effort to the specification of standard conferencing solutions. They include the Framework for Centralized Conferencing (XCON Framework) [2], which defines a signaling-agnostic architecture, naming conventions, and logical entities required for building advanced conferencing systems. An XCON-compliant framework architecture comprises several protocols, including the Binary Floor Control Protocol (BFCP) [3], which is associated with all moderation operations for a conferencing session.

As depicted in Fig. 1, BFCP models the presence of floor participants who ask for access to the conference floor (e.g. audio and/or video) by sending messages to a central entity called the floor control server. The server itself does not make decisions on its own, but rather forwards requests to the floor chair, who acts as a moderator and is in charge of making decisions.

When a conference participant asks for the floor, it sends a request message to the floor control server, which forwards it to the floor chair. When the chair makes a decision, it informs the floor control server, which in turn

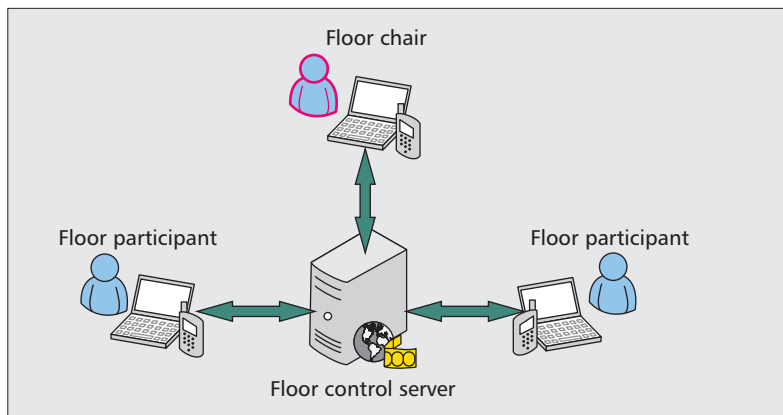


Figure 1. The Binary Floor Control Protocol architecture.

notifies both the requesting participant and all other participants potentially interested in receiving floor control notifications.

## RELATED WORK

Floor control has long been the subject of a number of research works. Before BFCP saw the light, some interesting approaches were proposed in the literature. The authors in [4] propose to use floor control in videoconferencing applications as a means to improve their scalability. With the proposed approach, moderation makes it possible to keep control over both processing and communication overhead, by allowing a maximum of two simultaneous streams at a time and hence mimicking a two-party videoconference. A different approach is embraced in [5]. Here, the authors propose an implementation of ALOHA and DQDB (Distributed Queue Dual Bus), two well-known MAC access protocols for local area networks, in a distributed overlay setting. The authors of [6] instead design an architecture for medium-sized peer-to-peer conferences. The system is equipped with a floor control mechanism to prevent too many users from speaking simultaneously (hence degrading the audio quality). To this purpose, a distributed role-based floor control protocol is introduced. The protocol leverages floor utilization statistics in order to optimize floor management activities.

More recently, the work in [7] has proposed an effective way to bring the functions made available by the BFCP protocol to the IP Multimedia Subsystem standard framework. They actually propose to use BFCP for the implementation of the IMS Fc interface envisaged by the 3GPP standard.

## UMPIRE ARCHITECTURE DESIGN

UMPIRE fills the role of floor chair in a conference. More precisely, when a conference starts, the UMPIRE user will log in as floor chair. Subsequent floor requests from conference participants will be:

- 1) Stored (in a PENDING state) in the First-Come-First-Served queue at the centralized floor control server.
- 2) Forwarded to UMPIRE, which will make decisions by assigning the floor to one or more users, thereby triggering state changes at the server. As an example, if the floor control policy

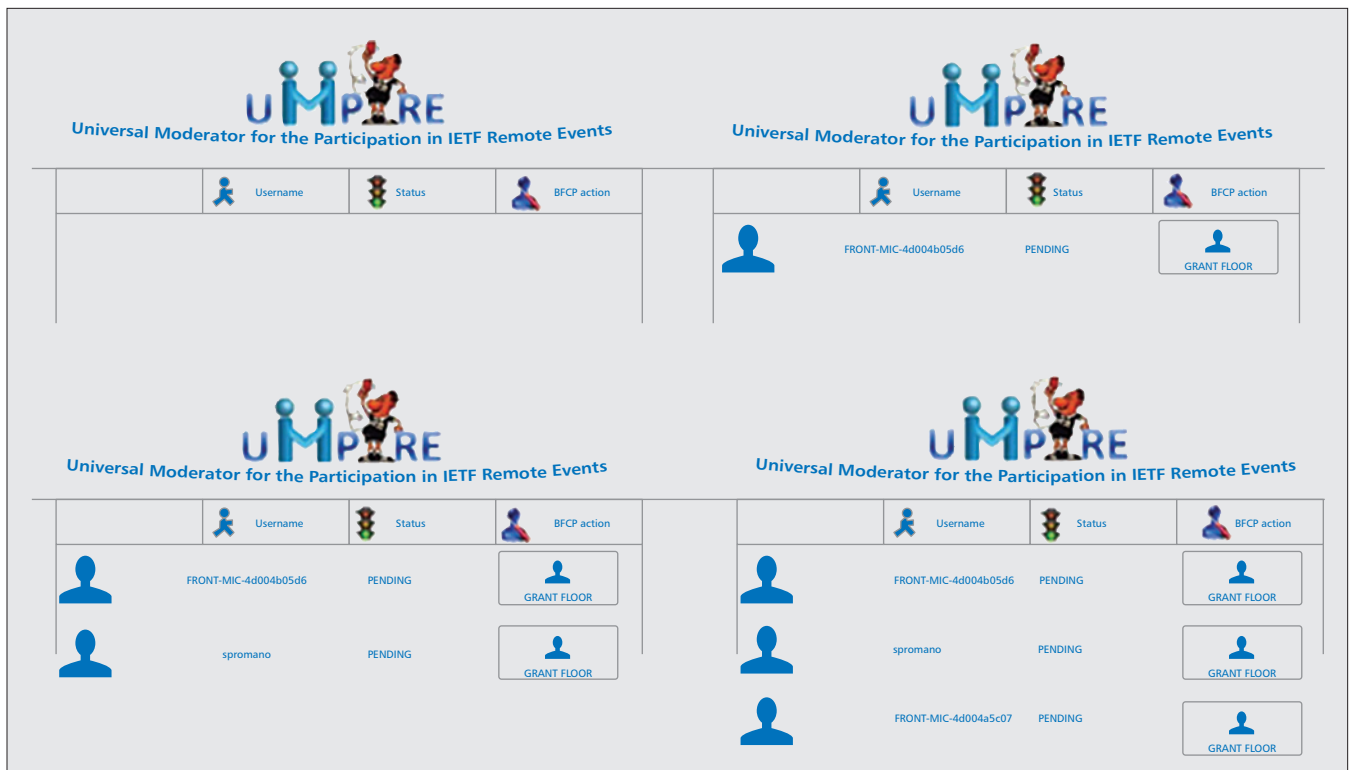


Figure 2. Two local participants and a remote participant asking for the floor.

has been configured to grant the floor to one user at a time and the UMPIRE accepts, in sequence, three requests coming from three different users, the following will happen:

- The first accepted request will move from a PENDING state to a GRANTED state.
  - The second and third requests will move from a PENDING state to an ACCEPTED state, indicating that they are ready to be served as soon as the currently GRANTED request has terminated (through either a floor release action from the client or a revoke action from the UMPIRE himself).
  - When the GRANTED request is completed, the first ACCEPTED request in the queue is granted the floor, while the second one becomes the ready-to-be-granted request in the floor control server queue.
  - If the server policy allows for a maximum of  $n$  requests to be granted the floor at the same time, up to  $n$  clients will reach the GRANTED state and will share the floor in question (e.g. in an audio conference, they will all be allowed to talk at the same time by contributing to the audio mix produced at the conference server).
- 3) Acknowledged to the clients through BFCP notifications (which allow participants to be kept up to date with respect to the state of their floor transactions).

The above scenario relies on the presence of a centralized floor control server with a queue managed by the chair and containing all floor requests coming from conference participants. This simple model allows for the introduction of advanced moderation functionality.

The interesting thing about the model is that the BFCP queue can be populated with requests

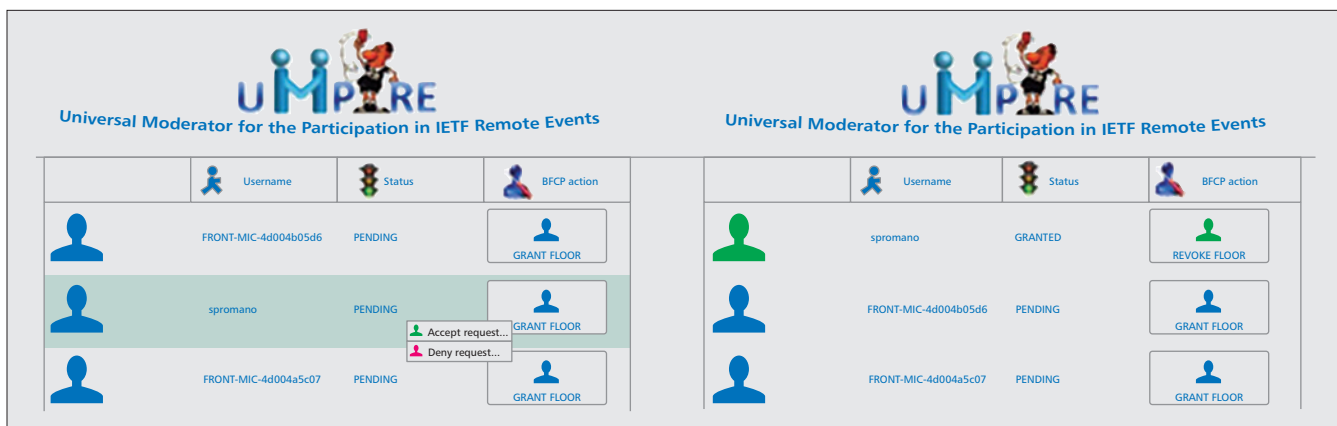
coming from both remote participants (equipped with BFCP-enabled clients) and local participants, thanks to the utilization of an agreed-upon procedure for requesting access to the conference floors when one is physically present in the conference room. The usual way of gaining the audio floor at regular IETF meetings is for local participants to politely wait (in a First-Come-First-Served queue) at one of the conference room microphones, in order to either ask for questions or provide their own view on the topic being discussed. If the microphones themselves were equipped with some simple means for:

- Recording the presence of users.
- Sending a trigger (i.e. a floor request) to the floor control server every time a new user lines up at the microphone.

The floor control server queue could transparently (and democratically) moderate a conference envisioning the contemporary presence of both local and remote participants. Indeed, this is what we implemented. With respect to the means for recognizing the presence of users lining up at the microphone, we decided to rely on the RFID technology. We opted for the following policy:

- An RFID reader was placed close to each of the conference room microphones.
- An RFID tag was assigned to each local participant willing to actively participate in the mechanism.

With this approach, when a local participant wanted to contribute to the ongoing discussion, all they had to do was to let their RFID tag be read by the RFID reader associated with the microphone at which they lined up. As soon as the participant's tag was read, the reader would send a BFCP floor request to the conference chair and let the participant be inserted in the centralized BFCP queue.



**Figure 3.** Accepting floor request coming from user “spromano.”

An image of the BFCP queue, updated in real-time, was always projected on a screen available in the room, for inspection by meeting participants. The chair of the conference (i.e. the UMPIRE user) was allowed to manage the BFCP queue by making decisions to grant the floor to individuals. This demonstrated straightforward moderation functionality, for a conference involving both remote and local participants.

## UMPIRE IMPLEMENTATION

UMPIRE has been implemented as a Web2.0 application, that is, a dynamic, highly interactive, web-based system. It is based on a bidirectional HTTP communication channel between the UMPIRE participant, acting as the floor chair in a conference, and the floor control server. The channel uses the COMET Server Push approach, as made available by the ZK framework.<sup>6</sup> Notifications from the server asynchronously arrive at the UMPIRE client and are represented on a web page providing an always up-to-date snapshot of the BFCP queue (with client requests and related BFCP states). Proactive actions undertaken by the UMPIRE (e.g. accepting or denying a PENDING request, or revoking a floor currently GRANTED to one of the participants), are immediately communicated to the floor control server, affecting the BFCP queue, as well as trigger floor notifications that are sent to clients.

### SAMPLE CALL FLOW

The UMPIRE functionality is demonstrated by the following simple scenario:

- Three users participate in a conference room:
  - Two local users, equipped with RFID tags:
    - \* User1: 4d004b05d6
    - \* User2: 4d004a5c07
  - A remote user (whose nickname is “spromano”), who enters the conference through a BFCP-enabled client.
- The order in which the three users ask for the audio floor is the following:
  - User1 → spromano → User2

The situation described above is illustrated in Fig. 2, which shows, respectively, the initially empty queue (top left), the first request arriving from “User1” (top right), the request from the remote participant “spromano” (bottom left), and

the final request from local user “User2” (bottom right). The final state of the BFCP queue, after all these actions have been performed, shows the three users (in order of arrival) in a PENDING state, that is, waiting for the chair to take actions.

UMPIRE first decides to grant the floor to “spromano,” as shown in the two snapshots in Fig. 3 (“Accept,” left snapshot), which translates into the following actions:

- The BFCP queue is modified: “spromano” becomes first.
  - The state of the BFCP queue is modified: the audio floor is GRANTED to “spromano” (right snapshot).
  - Remote user “spromano” is unmuted.
- UMPIRE now decides to grant the floor also to “User2.” This is illustrated in Fig. 4.

We can observe from the pictures that the following things have happened:

- UMPIRE has accepted the request issued by “User2” (left snapshot).
- The state of the BFCP queue changes: “User2” passes from PENDING to ACCEPTED (middle snapshot) and eventually to GRANTED (right snapshot). This lets us understand that the conference in question has been configured to allow multiple users to be granted the audio floor at the same time. Were this not the case, ‘User2’ would have moved from PENDING to ACCEPTED and would have stayed in such a state as long as the floor was held by “spromano.”

UMPIRE now decides to revoke the floor previously assigned to “spromano.” This is shown in the snapshots in Fig. 5, associated, respectively, with the action undertaken by the chair (left frame) and the effect it has on both the BFCP queue at the server and the web interface (right frame), which now reports “spromano” in red, with the related REVOKED status.

## LESSONS LEARNED

UMPIRE can be regarded as an advanced chapter of the author’s experiences with complex conferencing environments, such as the IETF, dating back to 2005. This chapter represents a clear example of the way research activities can be brought to the real world, if a proper engineering approach is embraced. The work done to develop the moderation platform taught us several lessons.

<sup>6</sup> <http://www.zkoss.org/>

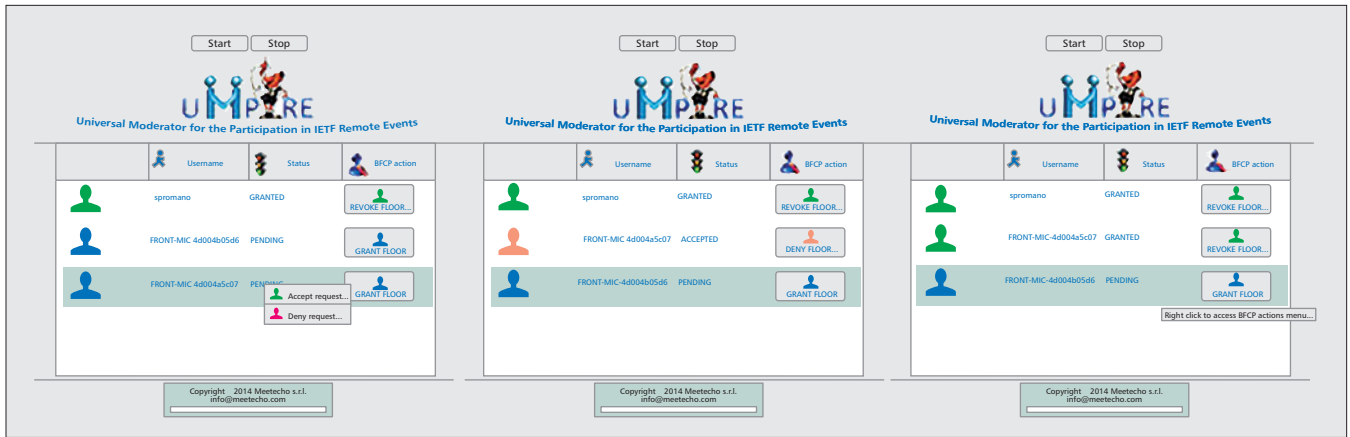


Figure 4. Accepting the request coming from the “second” local participant.

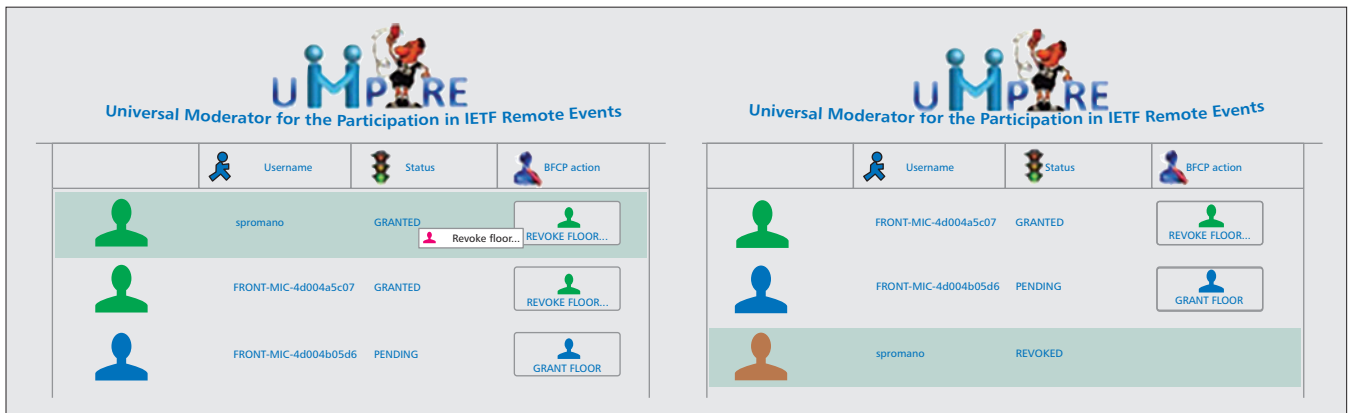


Figure 5. Revoking the floor to user “spromano.”

From the design perspective, our experience demonstrates the benefits in using a “separation of concerns” pattern for re-using a specific subset of the functionality in a large and complex system. Originally we were conducting research on the scalability of conferencing frameworks, and hence seized the opportunity to contribute to the ongoing standardization work within the IETF, by actively participating in both the Centralized Conferencing (XCON) and the Media Control (mediactrl) working groups.

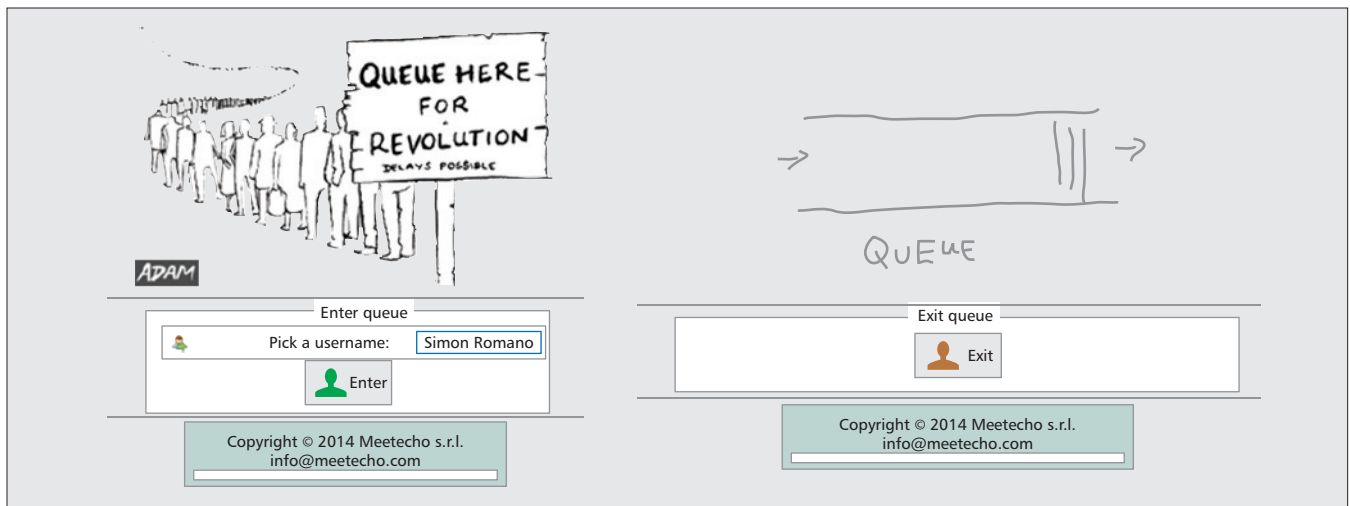
The eventual result was Meetecho, a spin-off of the University of Napoli offering a standards-based conferencing system used to support remote participation in IETF meetings. UMPIRE is built upon Meetecho’s major component dealing with BFCP moderation, which was implemented by our research group as a joint activity with Ericsson Nomadic Lab in Helsinki. Notably, it has been conceived as an independent component, which can interoperate with any other BFCP-compliant system; Meetecho is not required. Hence, we could focus solely on the client-side of the architecture. Besides re-using the code, we were also able to take advantage of the moderation server’s performance, which we have studied in detail in our previous work [1].

As the BFCP chair, UMPIRE is in charge of managing requests arriving from conference participants, while keeping an up-to-date representation of the queue, for the web-enabled GUI. The GUI itself was the most challenging part of

the overall prototype, since it represents a typical example of a bidirectional, HTTP-based component and must be capable of managing input events coming both from the web interface (e.g. when the moderator clicks on a user’s icon in order to perform a specific moderation action) and the server-side counterpart residing on the Meetecho conferencing server. For this part, we initially decided to use long polling HTTP requests sent by the client and responsible for the asynchronous update of the web view. This solution proved to be far from optimal, due to the unavoidable overhead of this approach. We then moved to a Comet server push approach, as described earlier. This is definitely better than polling when an application needs low latency events delivered from the server to the browser. Instead of repeatedly polling for new events, Ajax applications with Comet rely on a persistent HTTP connection between server and client.

Also worth mentioning is the communication between the RFID readers (which we install close to the room microphones) and the BFCP server. This part of the system is critical, since it raises both hardware and software issues. The reader has to interface with passive RFID tags (such as those “embedded” into conference badges worn by participants). It also has to properly communicate with the moderation server, by acting as a standard BFCP participant. Ultimately we used an effective, low-cost, programmable RFID reader made available by Phidgets Inc.,<sup>7</sup> a Canadian company offer-

<sup>7</sup> www.phidgets.com



**Figure 6.** The simple web GUI for sending floor request and floor release messages.

ing solutions for the rapid prototyping of RFID sensing components. Fortunately they come with built-in support for an application programming interface written in a number of different languages and freeing developers from all low-level issues associated with RFID sensing capabilities, so the programmer can focus on application-level issues. In our case, we leveraged the Java API.

As a final remark, we also notice that the idea of enabling RFID-based conference moderation continues to represent a critical aspect of the UMPIRE system. As part of the feedback we received after the first experiments at recent IETF meetings, we realized that many people would actually prefer a more lightweight approach to the interaction of conference participants (both local and remote) with the floor control server. Other participant signaling mechanisms are indeed possible, such as through personal smartphones. This suggested additional experiments to consider, with the final goal to allow the possibility that an operational service would permit multiple means for participants to signal their desire for the floor. Based on the previous consideration, we recently made a further developmental step and implemented a simple cross-device (laptop, mobile, tablet, etc.) web-based floor control client that can be leveraged by conference participants to send “RFID-less” floor request and floor release messages to the floor control server. With this extension, the UMPIRE system can now moderate a unified “virtual” queue that groups together local and remote participants who make use of either the RFID mechanism, or a BFCP-enabled conferencing application like Meetecho, or the simple Web GUI showed in Fig. 6. The enhanced version of the UMPIRE system has been used at the 90th IETF meeting in Toronto (July 2014).

## DISCUSSION AND CONCLUDING REMARKS

We have presented UMPIRE, a system for the automated management of floor control and moderation in a meeting room supporting the contemporary presence of local and remote par-

ticipants. At the time of this writing, UMPIRE (which was first proposed at the 83<sup>rd</sup> IETF meeting in Paris, in March 2012) has not yet been used ‘in the wild’ as a system to moderate actual meeting sessions. However, at recent meetings it has been demo-showed during the official “Meetecho tutorial for participants and WG chairs” and has gathered consensus and appreciation, besides stimulating useful feedback from the audience.

We are currently fine-tuning UMPIRE’s functionality to make it ready for official adoption within the IETF. Obviously, nothing prevents UMPIRE from being used in contexts other than the IETF. In fact, it can be employed wherever a plea for moderated access to shared resources exists.

## REFERENCES

- [1] A. Amirante *et al.*, “On the Seamless Interaction Between WebRTC Browsers and SIP-Based Conferencing Systems,” *IEEE Commun. Mag.*, vol. 51, no. 4, Apr. 2013, doi: 10.1109/MCOM.2013.6495759, pp. 42–47
- [2] M. Barnes, C. Boulton, and O. Levin, “A Framework for Centralized Conferencing,” RFC5239, June 2008.
- [3] G. Camarillo, J. Ott, and K. Drage, “The Binary Floor Control Protocol (BFCP),” RFC4582, Nov. 2006.
- [4] J. Garcia-Luna-Aceves *et al.*, “Floor Control Alternatives for Distributed Videoconferencing over IP Networks,” *Proc. Int’l. Conf. Collaborative Computing: Networking, Applications and Worksharing*, 2005.
- [5] S. Banik *et al.*, “Distributed Floor Control Protocols for Computer Collaborative Applications on Overlay Networks,” *Proc. Int’l. Conf. Collaborative Computing: Networking, Applications and Worksharing*, 2005.
- [6] Y.-J. Joung and P. H. Chien, “P2pconf: A Medium-Size P2P Internet Conference with Effective Floor Control,” *Proc. Int’l. Conf. Information Networking, ICOIN 2008*, Jan. 2008.
- [7] M. Al Rubaye *et al.*, “A Novel Architecture for Floor Control in the IP Multimedia Subsystem of 3G Networks,” *Proc. 69th IEEE Vehic. Tech. Conf., 2009, VTC Spring 2009*, Apr. 2009.

## BIOGRAPHIES

SIMON PIETRO ROMANO is an associate professor in the Department of Electrical Engineering and Information Technology (DIETI) at the University of Napoli. He teaches computer networks and telematics applications. He is also the cofounder of Meetecho, a startup and University spin-off dealing with WebRTC-based unified collaboration. He actively participates in IETF standardization activities, mainly in the real-time applications and infrastructure (RAI) area.