Technical Report No: 2000/05

Lyceum: the system and its architecture

Lucia Rapanotti
Jon G. Hall

2000

Department of Computing
Faculty of Mathematics and Computing
The Open University
Walton Hall,
Milton Keynes
MK7 6AA
United Kingdom

http://computing.open.ac.uk

The Open University

Lyceum: the system and its architecture

Lucia Rapanotti* and Jon Hall
Computing Department
The Open University
UK

Abstract

Lyceum is an audio and visual conferencing system for the Internet. Using Lyceum participants can exchange audio messages and share visual applications in real-time. The system is under development at the Open University to be deployed on distance learning courses. This document describes the system and its architecture.

1 Introduction

The Open University is the primary provider of distance learning in the UK. Distance learning is necessarily flexible in catering for the needs of students at a distance; the Open University has traditionally used primarily asynchronous modes of communication which do not constrain a student's interaction. Recent changes in the marketplace, and the ubiquity of the web, have necessitated and permitted the rethinking of this philosophy, with the result that courses with primary needs for synchronous communication can be presented.

Examples of such courses are:

- modern languages: in which conversation and interaction are paramount;
- software engineering: in which, for instance, formal technical reviews are an essential part of a student's training, but which are not possible using only asynchronous modes.

Other uses include group working in which asynchronous communication has been used successfully, but which do benefit from a synchronous model.

The Lyceum project is a product of research within the Open University on telepresence in distance education ([3]) and voice conferencing for students of foreign languages ([4]). Lyceum is an application providing synchronous communications to meet the needs of distance learning students. Lyceum works over the Internet. Using the system students can interact in real-time and share visual components to perform collaborative learning activities.

Lyceum entered its production phase at the beginning of 1999. A first release of the system is now in live use with Open University students and tutors. Development of the Lyceum system continues within the Open University's Learning and Teaching Services.

*Previously a member of the Learning and Teaching Services of the Open University.

2 Overview of the system

Lyceum is an audio and visual conferencing system developed for the Internet. Lyceum is developed using state-of-the-art component technology and is based on the client/server model. Both client and server contain pluggable componentry. The system is developed in Java, but relies upon a proprietary COTS technology for voice conferencing.

The component-based development of the system provides both technological and economical advantages:

- the COTS speech conferencing platform used in Lyceum can be easily replaced, so that Lyceum is not rendered obsolete as upgrades and new technologies become available;
- as customer requirements are many and various, the system is adaptable to a range of components, with short lead-in times equivalent to the turnaround time for a course;
- the system is recofigurable without the need for manual upgrade and/or re-compilation client-side.

Users access Lyceum from home with their PC connected to the Internet through a dial-up service, hence Lyceum's mode of operation is constrained by end-user computer specification and limited communication bandwidth. The Lyceum client has been designed to run on mid-range computer: the recommended minimun specification is a Pentium II at 266MHz, with 64 Mb of RAM, running Windows '95/'98/NT4.0, and connected through a 28.8 kbps modem. The COTS audio technology used in Lyceum provides very low-bandwidth transport of voice using state-of-the-art encoding. The technology is optimised for relatively low-speed exchange, such as modems at 28.8 or 33.6 kbps, and works with the *de facto* standard desktop sound subsystem (microphone and speakers or headset, plus any 16-bit Soundblaster-compatible sound card).

Lyceum was conceived as a tool to enhance the quality of the learning experience for students on distance learning courses by allowing real-time interaction with their tutors and other students. In particular, Lyceum is intended to be used by groups of students participating in real-time collaborative learning activities. Students can meet in virtual rooms created within the system and talk to each other while manipulating shared visual components.

As Lyceum will be deployed across a variety of university disciplines and courses, custom pluggable applications can be designed to suit a variety of pedagogical needs, and used within the system. Three shared visual components are currently available within Lyceum: a whiteboard, a concept map editor and a screen grabber. Also, the Lyceum client can be dynamically configured to use different combinations of visual components. In particular, components configurations and their updates can be downloaded by the Lyceum client from the Lyceum server at connection time. Finally, Lyceum comes with a moderation framework that can be configured to accomodate a variety of session control and moderation models. The moderation framework within Lyceum is very flexible and represents a distinguishing feature of the system. It is described in detail in the following section.

Lyceum has been tested quite extensively. A number of trials have been organised by the Open University Centre for Modern Languages with tutors and students of French and German, each trial involving about fifty users.

A more substantial test of the system is represented by M823 *Managing Knowledge*, a course part of the MBA programme at the Open University Business School. In its 2000 presentation, the course makes use of Lyceum for tutorials and assessment work. Over nine hundreds users are currently registered with Lyceum for this course, with up to two hundreds students on line at any one time.

3 The moderation framework

The Lyceum moderation framework is based on a set of policies and mechanisms which support session and floor control within the conferencing system. (The reader should refer to [2] for a comprehensive overview of the basic principles of session and floor control for multimedia conferencing and collaboration.)

The moderation framework is a distinguishing feature of Lyceum and its design has been strongly influenced by the pedagogical needs of the system's user community. The framework is:

- non-prescriptive, that is, it doesn't impose a particular moderation style, but is able to support a variety of moderation styles to meet the diverse users' needs;
- adaptive, that is, it allows moderation parameters to change dynamically. This is useful in order to support diverse pedagogical exercises within the same session.
- open, that is, based on an extensible programming interface. New moderation needs can be accomodated within the framework when, e.g., new visual components are added to the system or new pedagogical exercises are required.

The basis for the Lyceum moderation framework is discussed in the following sub-sections.

3.1 User levels

User levels define categories of users within the system, e.g., students, tutors and administrators.

User levels are expressed as numerical values and provide an indication of the user's *power* for each category of users. Such values are used in the framework to establish the privileges users have within the system. Higher values mean higher power. For instance, uses levels of 10, 20 and 30 for, respectively, students, tutors and administrators, indicate that students are the least powerful users and the administrator the most powerful.

The framework doesn't assume any particular number or range of values for user levels, and can then be configured differently for separate user groups.

3.2 Privileges

A user's privileges are things that the user is able to do within the system. Privileges result from combinations of values of *attributes* and *permissions*.

An attribute defines a property of a particular entity within the system, e.g., the user's (unique) ID within the system. User levels are attributes.

A permission expresses the minimum user level for a privilege. Permissions are expressed numerically as user levels - a permission with value 20, for instance, means that only users with user level greater than or equal to 20 are granted the corresponding privilege.

Attributes and permissions are often coupled. For instance, the framework defines a room's attribute `permanent`, which expresses whether a room resides permanently on the server (described later) or can be removed after a certain period of time. The framework also defines a permission `permission-setPermanent`, which indicates the minimum user level to make rooms permanent, i.e., to change the attribute `permanent` of a room.

Most of the attributes and permissions in the framework are *dynamic*, i.e., their value can be modified during operation. *Static* attributes are mainly user attributes, such as the user's ID and level.

Lyceum supports dynamic attributes and permissions through:

- the client's graphical user interface, which allows certain users to inspect and modify the current value of attributes and permissions;
- notification mechanisms between client and server, to propagate value changes;
- an *adaptation policy*, which rules the way the system adapts in response to changes of attributes and permissions.

Set-up values for attributes and permissions are kept in databases maintained by the Lyceum server.

The collection of attributes and permissions currently defined in the framework is quite extensive, and new attributes and permissions can be added at any time to define new privileges. A (non-exhaustive) list of the privileges currently defined includes: visibility privileges, which rule the visibility of rooms and user information; creation privileges, which rule the creation of rooms and setting of their properties; and policing privileges, which rule the assignment of floor control.

3.3 Rooms and room groups

In Lyceum, conferences are seen as *rooms*. The moderation framework assumes that rooms are organised into *room groups* and that room groups are arranged hierarchically. For instance, a possible configuration of the system for a distance learning course may have the structured depicted in Figure 1. In the figure, the root group `/` contains a plenary room called `HelpDesk` intended for users who may have troubles using the system. Under this, group `Course_A` contains two course rooms `/Social` and `Study`, where students and tutors on `Course_A` can meet. Then there are two sub-groups, `Sams_Group` and `Lyns_Group` for students tutored by Sam and Lyen, respectively. Each of these groups has rooms `Welcome`, `Team_1` and `Team_2`.

In general, users need not be concerned with this hierarchical organisation. In particular, all users need to know is the list of rooms that they can access. For instance, in the example of Fig. 1, students belonging to the group tutored by Sam would be presented by the client's GUI with a list of rooms including the three rooms in Sam's tutor group, plus the course rooms `Social` and `Study`, and the plenary room `HelpDesk`.

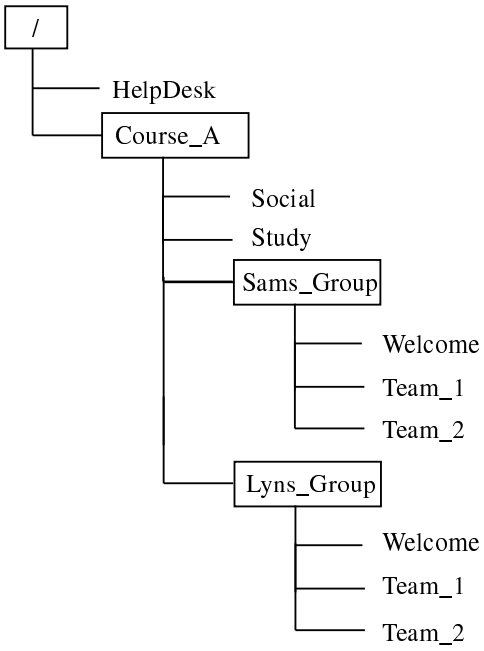

Figure 1: Example of rooms and room groups.

In the framework, the two views are reconciled as follows. Each user is assigned a *home* room group, that is a group to which the user belongs by default. In our example, all students tutored by Sam will be assigned home group `/Course_A/Sams_group`. By default, a user can access all the rooms in his/her home group as well as all the rooms in groups which are ancestors of his/her home group. This is the minimal set of rooms which are visible to a user. Permissions can be set so that users can visit rooms in sibling groups: for instance, we may want to allow tutors to access rooms in all the tutor groups.

The main reason why room groups are organised hierarchically is convenience, as the hierarchy allows for simple algorithms to set and propagate permissions. Also the hierarchy is intended to be rather flat, as it is in Figure 1. As a rule of the thumb, when setting up a conference, first it is important to identify the groups which are the leaves of the tree, as these are the groups which are most meaningful to the users. For instance, in the example, all the leaves are tutor groups under distinct tutors. Their parents groups should be seen as repository for rooms which are common to the users of all the subgroups. Indeed more complex hierarchical structures are possible, and the current framework could be extended to deal with them.

3.4 Moderation policy

The framework's moderation policy includes rules and assumptions upon which the framework is based and governs the way privileges are granted to users based on attributes and permissions. Several rules make up the current moderation policy of Lyceum. For instance, the following rule defines how privileges propagate among rooms and rooms groups: "Values of attributes and permissions propagate through the group hierarchy, from groups to their rooms and children groups. Values lower in the hierarchy override values further up the hierarchy."

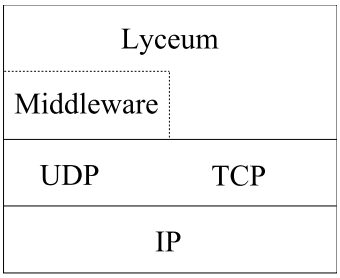

Figure 2: IP model view of Lyceum.

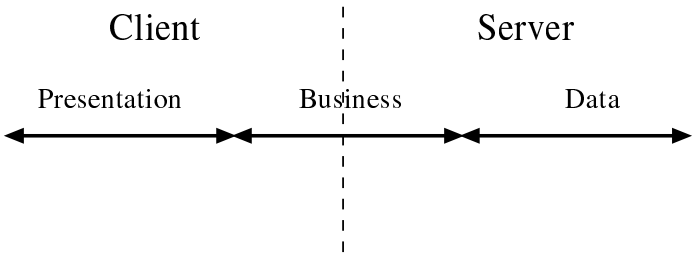

Figure 3: 3-tier model view of Lyceum.

4 The architecture

Lyceum is a conferencing system for the Internet. Figure 2 illustrates a layered view of Lyceum from an IP model viewpoint. Lyceum sits at the application layer and makes use of the IP protocols TCP and UDP: TCP, directly, for the exchange of visual and control data among client and server; and UDP, through its COTS audio component, for voice conferencing.

Lyceum is a 3-tier distributed application as indicated in Figure 3, with presentation at the client side, data management at the server side, and business rules distributed between client and server.

The following subsections describe the Lyceum client and server in some detail.

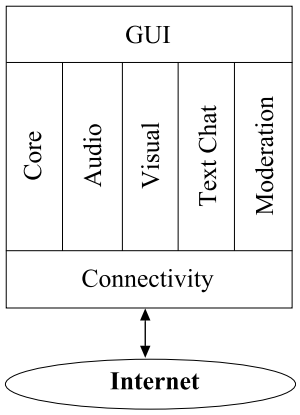

Figure 4: The Lyceum client.

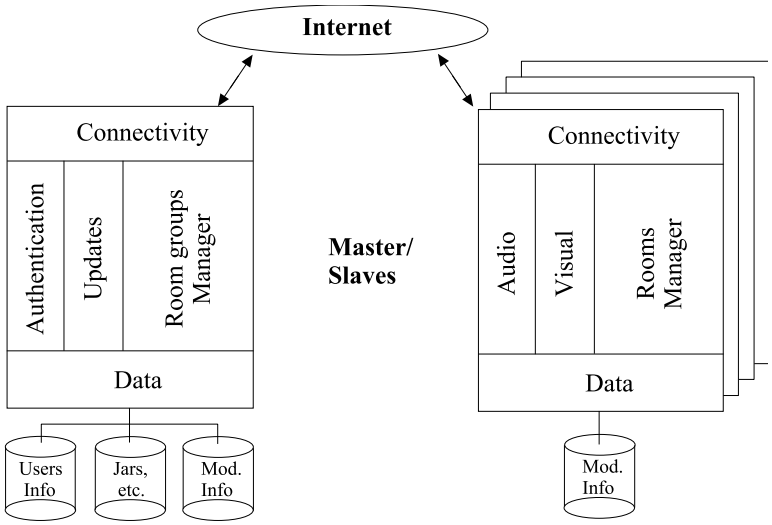

Figure 5: The Lyceum server.

4.1 The client

The client is organised as in Fig. 4. The application is launched from its *Core* component that starts up the client graphical user interface (GUI).

Through the client's GUI, users' requests are filtered down to a number of functional components. The *Moderation* component enables/disables client's components according to the moderation settings. The *Visual* component handles events generated within one of the shared visual components and exchanges related visual events and control signals with the server. The *Audio* component handles audio events and voice communication with the server. The *Text Chat* component manages an asynchronous text communication channel among participants of a conference.

4.2 The server

The server has a master/slaves architecture and, functionally, is organised as in Fig. 5.

The master, on the left of the figure, offers two main services. The first one includes *Authentication* of users and *Updates* of software components. For user authentication, the server has access to a user database. Components updates are negotiated with the client: when a client connects to the server, they exchange information on the current status of the client's visual components; if necessary new code is downloaded from the server, which may be a new visual component or upgrades of existing ones. Message-of-the-day and other user information are also transferred from the server to the client at connection time.

The second service offered by the master is the management of room groups through its *Room Groups Manager* (RGM). The RGM has responsibility for creating and maintaining room groups, distributing rooms among the slave servers, and assigning users to rooms at connection. The RGM also maintains and propagates some of the moderation information. Moderation information is uploaded by the server at start-up from a database.

For efficiency gains, the two services of the master server are carried out by independent processes (more about this later).

Slave servers handle a number of rooms, which are assigned to them by the RGM in the master server. For each room, through a *Rooms Manager*, they maintain the state of the room (including its users list), the status of its audio and visual components, and maintain and propagate moderation information. The other services offered by each slave server mirror the services offered by the client application, that is handling audio, visual and text chat events and related control communication.

There is no restriction on the number of slaves servers that the master can handle, other than as imposed by the capability of the host's operating system to handle IP connections. In one of its current pilots, the Lyceum system is configured to maintain approximately three hundred and fifty permanent rooms, which are distributed among three slave servers. Temporary rooms dynamically created by conference participants are allocated by the master server to its slaves in a round-robin fashion.

4.3 Workflows

The Lyceum client handles the connection to the server using the followign three-step procedure:

1. when the client connects to the server, first it uses the server's authentication and updates service. If positively authenticated, the client is sent some user related moderation information retrieved from the user database;
2. then the client connects to the RGM, which decides which room the client should be connected to, and returns to the client this information together with the location (IP address and port number) of the slave server managing that room. The client also receives a list of all the rooms that it is allowed to access within the system. This is established by the RGM by applying rules in the moderation policy;
3. finally, the client connects to the slave server where its default room resides. At this point the user can take part in the conference.

After logging onto the system, the client has knowledge of, and can communicate with, the master server, its RGM and the slave servers that manage all the rooms which are accessible by the client.

The client will send requests concerning room groups' moderation information to the RGM, as well as requests for the creation of new rooms. The client will send requests concerning its current room's moderation information to the slave server. Audio, visual and control communications will also be exchanged with that server. A room change is handled by the client as a disconnection from the current slave server followed by a connection to the slave server of the room the user wants to join.

Notice that the responsibility for supporting the moderation framework is shared between the server(s) and client applications. In particular, both the RGM and the rooms managers in slave servers have responsibility for maintaining and propagating moderation information, while the client receives such information and applies the framework's moderation policy.

Dynamic changes to moderation attributes and permissions occur through the client interface. Such changes are communicated to the server (both master and slaves, depending on what piece of information is being changed), which notifies

the changes to all parties concerned. Client applications adapt their behaviour as a result of such notification and according to the framework's adaptation policy.

4.4 Architecture rationale

Behind the current architecture of Lyceum are issues of perfomance, scalability, and fault tolerance.

The services offered by the master server are very light. Authentication, updates and routing of clients to their slave servers are carried out by two independent processes and constitute a minor part of the server's overall workload, hence they can be carried out very efficiently. Most of the workload is represented by audio and visual traffic within rooms, which are distributed among slave servers. Such workload has to be balanced among slave servers in order to guarantee uniform service for the users. In the general case this can only be achieved if the distribution is based on an accurate estimate of such workload. In our trials, simple and accurate estimates are possible because of the way students conferences are set up, and a simple round-robin algorithm is adopted by the RGM in order to load balance the slave servers.

The Lyceum server is highly modular, which has a positive impact on its scalability. Not only services are provided by multiple slave servers, but each slave server makes use of a number of separate COTS audio componenst. This means that the system scales up by increasing both the number of slave servers and the number of audio components per slave server. In a current pilot of Lyceum the system is configured to use three slave servers, each relying upon five separate audio components.

In various trials of Lyceum and its underlying technology, it was noted that the audio conferencing subsystem is the most fragile component, hence the most likely to go wrong. This is not unexpected as the audio subsystem is the one which has the highest real-time demands. The current architecture of Lyceum exhibits good fault-tolerance to audio failures through the use of separate COTS audio components per slave server, and a recovery policy in case any such component fails. In particular, each slave server distributes its load among a number of COTS audio components. The failure of one such component only affects rooms serviced by that component, which is usually a small proportion of the total workload of the system. The slave server can detect such a failure and recover by restarting a new audio component in no more than one second, so that disruption to conferences is kept to a minimum.

5 Work in progress

Further development and maintainance of the Lyceum system are the responsibility of the Open University's Learning and Teaching Services. Current development concentrates on the 'text chat' facility and the dynamic aspects of the moderation framework. The development of bespoke visual components to use within the system is customer-driven and handled by the development team as requirements arise.

Trials of the system have already generated valuable information on defects and usability issues related to the product, that have been fed back into the development process. The modular, component-based architecture of Lyceum has proved very effective in allowing the development team to rectify problems

and distribute upgrades without requiring recompilation and redestribution of the whole client application.

Data have also been collected on user problems caused by latency and poor reliability of the network connections between clients and server. The causes of such problems are not under the project's control, of course, and relate to more general issues of quality of service for real-time communication over the Internet. A more thorough investigation into the occurrences of such problems and their impact on the usability of the product would, however, be beneficial as well as adding adaptation strategies, client- and server- side, which would contain the ill-effects of such disruptions. The design of software frameworks for quality of service over the Internet is a young discipline [5, 1], and one that could prove of extreme benefits to real-time multimedia applications such as Lyceum.

Acknowledgements

Many people have contributed to the current design of Lyceum. The authors would like to thank the Lyceum development team for all their hard work in bringing the system to life. In particular, Sam Marshall, chief Software Designer of Lyceum, had lots of the ideas on the current architecture of the system and on most of its implementation. Mark Eisentstadt, Craig Rodine and Lesley Shield have all contributed in discussions which have influenced the design of the moderation framework.

References

[1] S.N.Bhatti, G.Knight, "QoS Assurance vs. dynamic adaptability for applications ", *Proceedings of NOSSDAV*, 1998.

[2] Hans-Peter Dommel, J.J. Garcia-Luna-Aceves, "Floor control for multimedia conferencing and collaboration", Multimedia Systems, vol. 5, pp. 23-38, 1997.

[3] "Stadium", Knowledge and Media Institute, The Open University, URL "http://kmi.open.ac.uk/stadium/".

[4] M.Kotter, L.Shield and A.Stevens, "Real-time audio and e-mail for fluency: Promoting distance language learners aural and oral skills via the Internet", ReCALL Journal, special Web-based edition, July 1999.

[5] X.Xiao, L.M.NI, "Internet QoS: a big picture", IEEE Network, vol. 13, no. 2, pp. 8-18, 1999.

A Moderation framework specification (version 0.1)

This appendix contains the current specification of the Lyceum moderation framework.

A.1 Attributes

A.1.1 User attributes

- `userName`: unique user identifier.
- `password`: password to connect to server.
- `realName`: user's family name.
- `tagName`: user's preferred name, that is the name the user would like to be know as during the conference.
- `level`: user's power level.
- `group`: path identifying the user's home group.
- `blocked`: indicates whether the user has been blocked.

A.1.2 Room group attributes

- `moderated`: boolean value indicating whether the rooms in the group are moderated or free.
- `audioOnly`: boolean value indicating whether the rooms in the group are audio-only.
- `nameType`: one of three values (`user|real|tag`), indicating the type of the names displayed the users lists of the rooms in the group.
- `showLevel`: boolean indicating whether users' level should be represented in the users lists of rooms in the group.
- `permanent`: boolean indicating whether rooms in the group are permanent.
- `lifeSpan`: number of hours non-permanent rooms of the group are kept after their creation.
- `blocks`: comma-separated list of all users blocked in the rooms of the group.
- `friendlyName`: friendly name of the group to be used in user interface.
- `defaultRoom`: the default room of the group.

A.1.3 Room attributes

- `moderators`: comma-separated list of all the users in the room with moderator status.
- `moderated`: boolean value indicating whether the room is moderated or free.
- `audioOnly`: boolean value indicating whether the room is audio-only.
- `nameType`: one of three values (`user|real|tag`), indicating the type of the names displayed the room's user list.

- `showLevel`: boolean indicating whether users' level should be represented in the room's users list.
- `permanent`: boolean indicating whether the room is permanent.
- `lifeSpan`: number of hours a non-permanent room is kept after its creation.

A.2 Permissions

A.2.1 General

- `permission-administration`: user level required to perform administrative tasks.
- `permission-visitSiblingGroup`: user level required to visit a group sibling to the user's home group. (As home groups are always leaves of the tree, this applies only to lower level groups.)
- `permission-createRoominInSiblingGroup`: user level required to create a room in a group sibling to the user's home group.
- `permission-controlFloor`: user level required to have floor control in a room.

A.2.2 Room group permissions

- `permission-createRoom`: user level required to create new rooms within the group.
- `permission-setPermanent`: user level required to make rooms within the group permanent or not.
- `permission-changeModules`: user level required to select a visual component (module) in the rooms within the group.
- `permission-setModeration`: user level required to make rooms within the group moderated or not.
- `permission-setModerator`: user level required to give a user floor control of a moderated room within the group.
- `permission-viewUser`: user level required to view information about another users, such as their real name and user name (and level), above what shown in the users lists of the room within the group.
- `permission-setNameType`: user level required to alter the type of names shown in the users list of rooms within the group (i.e., to set the nameType attribute).
- `permission-setShowLevel`: user level required to control whether indication of user level is shown in the users lists of rooms within the group (i.e, to change `showLevel` attribute).
- `permission-setAudioOnly`: user level required to set the audio-only status of rooms within the group (i.e., set `audioOnly` attribute).
- `permission-setBlocks`: user level required to block users (i.e., set `blocks` attribute) in rooms within the group.

A.2.3 Room permissions

- `permission-setPermanent`: user level required to make the room permanent or not.
- `permission-changeModules`: user level required to select a visual component (module) in the room.
- `permission-setModeration`: user level required to make the moderated or not.
- `permission-setModerator`: user level required to give a user floor control of the room when moderated.
- `permission-viewUser`: user level required to view information about another users, such as their real name and user name (and level), above what shown in the room's names list.
- `permission-setNameType`: user level required to alter the type of names shown in the room's users list (i.e., to set the nameType attribute).
- `permission-setShowLevel`: user level required to control whether indication of user level is shown in room' users list (i.e, to change `showLevel` attribute).
- `permission-setAudioOnly`: user level required to set the room's audio-only status (i.e., set `audioOnly` attribute).
- `permission-block`: user level required to block a user in the room (i.e., set `blocks` attribute).

A.3 Moderation policy

The moderation framework is based on the following rules:

R.1 Values of attributes and permissions propagate through the group hierarchy, from parent groups to their rooms and child groups. Values at the bottom of the hierarchy override values further up in the hierarchy.

R.2 Each user has a home group and each group has a default room. When a user connects to the system, the user joins the default room of his/her home group.

R.3 A user has visibility of all the rooms in his/her home group and all the rooms in ancestors groups of his/her home group.

R.4 A user can create a new room in any of the groups he/she has visibility of and in which `permission-createRoom` is less or equal to the user's power level.

R.5 A user can be given permission to visit room groups which are siblings of his/her home group (through `permission-visitSiblingGroup`) and to create rooms within such groups (through permission `permission-create RoominInSiblingGroup`).

R.6 Floor control only affects the visual component of a room. That is if a room is moderated then everybody can talk but only moderators can actively use the visual component. The only way to prevent users from talking is by blocking them.

R.7 In a moderated room, if at any one time no participants have floor control (i.e., there are no users nominated as room moderators or with `permission-controlFloor`), then the room behaves as a free room until a user with floor control joins the room or is nominated.

R.8 A moderator has the power of nominating other moderators or taking away their moderation status, transferring floor control to another user, and relinquishing floor control.

R.9 A moderator looses moderation status when leaving the room.

R.10 If a room is made free all its moderators loose moderation status.

R.11 Blocked users cannot broadcast audio or data. That is, block users can still join rooms, listen, and watch the visual component, but they cannot talk and cannot alter the visual component.

R.12 Blocking a user who has been nominated as the moderator of a room will remove his/her moderation status.

R.13 Users with `permission-controlFloor` are not affected by room moderation and user blocks. That is they can always take active part in the conference even in a moderated room, and they cannot be blocked.

R.14 Audio-only rooms cannot have selected modules. A room with a selected module can be made audio-only after deselecting the module. An audio-only room has to be turned into an ordinary room before a module can be selected.

R.15 Audio-only clients can only see audio-only rooms.

R.16 A non permanent room is destroyed when its life span expires. If at that time the room is not empty, then it is destroyed one hour after it has become empty.

R.17 All permissions are expressed as power levels. We adopt the convention that `permission-setXXXX` denotes permission to change attribute `XXXX`.

R.18 Administrators accessing the moderation framework though the server console can set all permissions and attributes. Administrators accessing the moderation framework through the client application can set all permissions and attributes except `permission-administration`. All other users can only access the moderation framework though the client application and can only change attributes if given the corresponding permissions.

R.18 The current value of all permissions and attributes is kept by the server. The server provides synchonisation mechanisms to guarantee mutual exclusive write access to the data.

R.19 Permissions and attributes are static if they cannot change while the user is on-line, dynamic otherwise. For dynamic attributes the client application provides suitable GUIs to view and set values.

Note that the list of modules per each room is not stored within the moderation framework. This information is available to client applications through the Lyceum client-server interface.

A.4 Adaptation policy

The adaptation policy rules the way the client and server applications adapt to values of permissions and attributes and their changes.

The server has to maintain the current configuration, provide suitable synchnonisation mechanisms for write access to shared data and backup the current configuration to permanent storage.

The client's behaviour is affected by the moderation framework as described in the following subsections.

From now on, we say that a user has `permission-XXX` if his/her user level is higher than or equal to the current value of `permission-XXX`.

A.4.1 When the user connects

If the user has `permission-administration` the client GUI will allow the user to access administrative tools to view and modify attributes and permissions.

A.4.2 When the user enters a room

- The name of the current room is displayed as the label of the 'rooms pull-down menu'. If the user has just connected to the server, the current room has to be determined via the user's `group` and the group's `defaultRoom`.
- Names are displayed in the users list according to the room's `nameType`.
- Indicators of user level are displayed or not according to the room's `show level` attribute.
- If the current room is `moderated` then all its `moderators` are indicated in the users list.

A.4.3 When the user clicks on the "room pull down menu"

A menu is displayed with the list of rooms visible to the user from the group the user is currently in up to the top of the group hierarchy. The menu also contains:

- If the user is in his/her home group:
  - the option "create room ..." if the user has `permission-createRoom`;
  - the option "go to group ..." if the user has `permission-visitSibling Group`;
  - the option "current room setting ..." if the user has any of the `permis sion-setXXXX`, i.e., any of the permission which allow the user to change an attribute.
- If the user is not in his/her home group then:
  - the option "create room ..." if the user has `permission-createRoom InSiblingGroup`;
  - the option "go to group ..." (to be here the user must have `permission -visitSiblingGroup`!)

A.4.4 When the user left clicks on somebody else's user name

This interaction has effect only if the user is in his/her home group.

If the user has `permission-viewUser`, the user name, real name, tag name and level of the other user are shown in a pop-up.

A.4.5 When the user right clicks on his/her name

This interaction has effect only if the user is in his/her home group.

- If the user is not a moderator, but has `permission-setModerator`, then a pop-up is displayed containing the option "add floor control".
- If the user is a moderator, then a pop-up is displayed containing the option "remove floor control".

A.4.6 When the user right clicks on somebody else's user name

This interaction has effect only if the user is in his/her home group.

- If the user has `permission-setBlock`, then:
  - If the other user is not blocked and doesn't have `permission-control Floor`, then a pop-up is displayed containing the option "block user".
  - If the other user is blocked, then a pop-up is displayed containing the option "unblock user".
- If the user is not a moderator, but has `permission-setModerator`, then:
  - If the name corresponds to a moderator then a pop-up is displayed containing the option "remove floor control".
  - If the name corresponds to a user with floor control, then no pop-up is displayed.
  - Otherwise, a pop-up is displayed containing the options: "add floor control".
- If the user is a moderator, then:
  - If the name corresponds to a moderator then a pop-up is displayed containing the option "remove floor control".
  - If the name corresponds to a user with floor control, then no pop-up is displayed.
  - Otherwise, a pop-up is displayed containing the options "add floor control" and "transfer floor control".

A.4.7 When the client is notified that an attribute has changed

This happens when an attribute of the current room has changed and the client is notified through the notification mechanism. The behaviour is described below for each attribute. Note that no notification is necessary for permissions as permissions only affects user interactions, hence they are checked when user interactions occur.

- `moderated`: the users list is updated accordingly. For instance if the current room becomes non moderated and there were moderators in the room, they loose their moderator status and are displayed in the list as ordinary users.
- `moderators`: if moderators are added/removed, their new status is reflected in the users list.
- `nameType`: the users list is updated to display the new name type for each user.
- `showLevel`: the users list is updated to display/hide the indication of user level.

A.4.8 When the user changes the value of attributes and permissions

When the user changes attributes and permissions through the client GUI, the client notifies the server, which updates the current configuration. This may involve resolving conflicts which arise from concurrent write access to share data. All the clients which are interested in the changes are notified through the notification mechanisms.