2. TANGO: Design Goals, Architecture, and Implementation.

2.1. Rationale

In spite of its wide acceptance, the traditional World Wide Web is little more than a convenient method of accessing and displaying information. Use of the stateless HTTP protocol greatly contributed to the success of the Web, but it also strongly limited its further development. The basic WWW model does not support any real interaction.. The introduction of the CGI (Common Gateway Interface) mechanism did not solve the problem. At this stage of the World-Wide Web development, it was impossible to create tools for cooperative work based on this technology. The situation has changed with the introduction of some new technologies.

In the 1995 Sun Microsystems, Inc. introduced Java (Gosling 1996). Among other well published features, Java introduces a notion of applets. Applets are programs that can be included into HTML pages, much like an image. A Java-compatible browser can be seen as an extensible container capable of securely executing chunks of the code distributed to the clients from an HTTP server. It is interesting to note that the idea of downloadable interpreted software is hardly new: in the late 80s, Sun introduced a product called NeWS which used extended Postscript as both programming language and communication protocol. The idea was apparently too far ahead of its time - NeWS project was promptly abandoned. The combination of Java and HTTP seems to carry much greater appeal. Java gained enormous popularity among the World Wide Web users. Together with its client side script partner, JavaScript, it is now broadly used to bring interactivity into the Web environment.

The notion of interactivity is somewhat fuzzy. Currently, an "interactive" Web application is the one that creates dynamic HTML pages dependent on user input. In this paper we are not concerned with this sort of interactivity. Rather, we describe a system that will allow human users to interact via their computers. Such systems are known under the newspeak term of *collaboratory*.

Computer mediated collaboratory systems have a relatively long history (for an extensive review, see Schooler 1996). Despite of the extensive body of research, collaboratory systems are hardly out of the laboratory. Academic implementations have been evolving together with the enabling multimedia and networking technologies: only recently the sophistication of the affordable desktop technology reached the level making multimedia collaboratory a viable prospect. The commercial collaboratory implementations are exceedingly primitive: video teleconferencing, shared whiteboard with limited graphical capability, and a textual chat tool represent industrial state of the art, as exemplified by Insoft's Communique!, SGI's In-Person, or Intel's ProShare products. It is not very surprising that this type of collaboratory is perceived by some as an activity in search of a need (Grudin 1990).

Why a Web-based collaboratory system? From a sociological standpoint, we believe that today's web is a prototype of the most common environment in which people interact with computers. If computer collaboratory is to become a commonplace, it must be Web based per definition. The overwhelming success of the Web creates new opportunities for collaboratory. From the technological standpoint, collaboratory systems critically depend on interoperability. Web is undoubtedly the most successful implementation ever of an open system. Web's very nature creates a coherent programming environment which is conducive for collaboratory experiments. Java, with its platform independence, is an ideal language for implementation of collaboratory systems.

Another important argument for a web-based collaboratory is the vast information contents of the Web. The limited success of the traditional collaboratory tools is at least partly attributable to their poor integration with information retrieval tools. With little content, collaboratory process becomes meaningless and artificial. Web collaboratory has a potential for seamless integration of information retrieval and exchange tools, with vast data repositories readily available.

We have designed and implemented a web-based collaboratory system, code-named TANGO, starting from the following few guiding principles that reflect our understanding of collaboratory concepts:

- *We do not really know how to build an efficient collaboratory*. Since no truly successful collaboratory system exists, there is no blueprint. We have to therefore allow for an extensible system with very few limitations. TANGO must not define application specific protocols, application programming language, or limit in whatever way functionality of collaboratory applications.

- *The essence of the collaboratory function must be defined by application and by application only*. It is up to application developer to specify requested collaboratory functionality for every application. This functionality may be obvious, as for a chat or audio conferencing, or highly non-trivial, as for a collaborative weather prediction and analysis application with real-time computational back-end. TANGO supports collaborative process for either type of application.

• *The most likely collaboratory applications are the applications supporting professional work in the stand-alone mode.* This implies a necessity for a simple API enabling fast and easy porting process of the existing applications into the collaborative framework. TANGO supports such an API for applications written in few languages. By providing such an API, TANGO fully benefits from the following simple observations:

1. the majority of the code of existing applications may be reused while building their collaborative versions
2. the majority of applications working in collaborative mode may be implemented using similar and actually simple message exchange mechanism
3. very useful distributed collaborative systems can be built by integration of slightly modified standalone applications together with some user and session management mechanisms

TANGO is not the only web-based collaboratory system. There are multiple ongoing projects in the area of Web based distributed systems. Some of them aim at development of a platform for implementation of distributed software in Java (NCSA Habanero 1996). Some of the others build software simplifying development of such distributed applications (Caltech Infosphere 1966). The majority of other projects concentrate on development of single-purpose distributed or collaborative systems themselves. Designing TANGO we tried to build a generalized, extensible, Web based collaboratory system capable of creation of diverse shareable information spaces. The system provides support for coordinated but independent views of related information streams, with capability for high-end visualization and interactive manipulation of multimedia information. TANGO extends Web paradigm to the domain of collaborative computing and well beyond the concept of the chat, shared whiteboard, and replicated, identical instances of simple generic applications. Tango is a generic tool used to share views.

2.2. System requirements

We have based design and implementation of the TANGO system on the following requirements:

- integrate both standalone and Web-based applications by providing a uniform interface to communicate with their instances on remote machines;
- allow to execute and control collaborative application from the Internet browser environment;
- provide means for session control (user authentication, starting and ending sessions, tracing participants activities, changing user privileges);
- enable integration of existing applications written in any programming language, assuming socket communication as the only necessary communication mechanism;
- provide ability to download certain applications across the network through the use of Java applet technology, allowing main parts of the system to be automatically distributed across the Internet;
- provide logging mechanism, so all user activities may be stored in the persistent form in a database and retraced if necessary;
- support definition of compatible message and application classes to enable multiple, task oriented views of the information streams

2.3. System overview

The main functionality provided by the system consists of the following elements:

- session management
- communication between collaborating applications (data and event distribution)
- user authentication and authorization
- event logging

2.3.1. Session management

A *session* is a group of application instances currently working together in collaborative mode. All applications belonging to the same session exchange information and share behavior. How particular application operates in collaborative mode depends on this application characteristics.

Each application belongs to a session, even if it is not currently used for collaboration. In such case the session consists of this one application only.

To provide for the floor control, for each application in a sessions there is one distinguished user which is considered to be a *master*. Master of the session has special privileges of controlling the application behavior and/or controlling access of other users to this session. The privileges depend on the application type. The master status is transferable

For multimedia applications using high-volume data streams (audio/video, hi-res images, 3D), session management interacts with the system layers controlling multicast capability.

2.3.2. Communication

Only applications belonging to the same session can communicate. A robust message passing system supports this communication. Message and event distribution proceed automatically under control of the session manager.

There are two major types of messages: control messages, and application messages.

Control messages are generated by the system for communication among server, demons, and control applications. These messages serve for logging users into the system, establishing sessions, launching applications, etc. Control messages are invisible for user applications.

Application messages are main means of communication among user applications. The structure of application messages depends on the application and is not interpreted by the communication system. The communication system is transparent for the application messages.

2.3.3. User authentication and authorization

System provides means for a multilevel user authentication and authorization. TANGO security mechanisms are independent on the underlying operating system.

2.3.4. Event logging

One of the important system capabilities is recording of the system activity. Since all system and application messages must go through the main server, all of them can be recorded in a database. Each time a control or application specific message passes the server this fact is recorded in the database together with the date, exact time, and sender information. These data can be then accessed and the whole system activity can be replayed.

Event logging is a basic feature supporting *asynchronous collaboration*. Note that the term of asynchronous collaboration is frequently used to describe e-mail and other messaging systems. In the collaboratory context, asynchronous collaboration denotes ability of the collaboratory system to replay or even summarize at a desired level the entire collaboratory sessions.

.

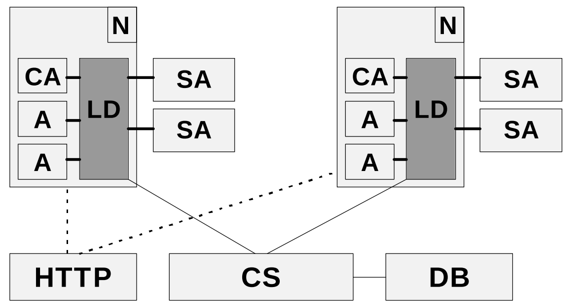
2.4. System architecture

2.4.1. System elements

Figure 1 presents the global architecture of the system. The system consists of the following components:

Local demon's main task is to maintain two way communication between user applications, applets and the central server and to lunch local applications. Demon is implemented as a plug-in in Netscape browser.

Central server is the main communication element. All local demons communicate with the central server. The server maintains the system state data. Its main tasks are routing of messages among applications participating in each session and recording all events in a dat abase.



CA - Control Application
A - Java Applet
SA - Standalone Application
LD - Local Demon
CS - Central Server
HTTP - HTTP Server
DB - Database
N - Netscape Navigator

Figure 1. Global architecture of the system

All user applications which run as standalone programs are call *local applications*. Local application may be written in any programming language. All local applications communicate with the local demon by the use of sockets. The demon is responsible for starting these applications and routing messages to and from applications.

Java applets are also user applications but written in Java language, downloaded through the network from an HTTP server, and executed in Netscape environment. Communication between Java applets and central server is also maintained by the local demons. Java applets communicate with local demon by calling its method functions.

Control application is a specialized application which acts as an interface to the user. Control application is used to launch applications locally or remotely, create and join existing sessions, exit applications, etc. This application allows also logging into the system. User interface to the control application depends on his/her privileges, giving a user access to only these features that he/she is allowed to use. Control application is customizable

2.4.2. TANGOsim extension

An example of TANGO architecture flexibility and extensibility is TANGOsim - an discrete event simulator. A multithreaded simulation engine implementing virtual time can be driven by either a scripting language or interactively controlled by a user via Simulation Controller. The engine and the controller are implemented as a Java application and a Java applet, respectively. The simulation engine can create messages for any application compatible with TANGO system, to create and control sessions, and to realize scenarios in which the course of action depends on user input. TANGOsim is a prime example of the system capability to go beyond the collaboratory model of "cooperating twins".

2.5. Implementation Issues

Ideally, for portability reasons, one we would like to implement all system components using only Java. This approach turns out, however, to be currently unfeasible for several reasons. The most important are security constraints imposed on Java.. Using pure Java model we would have to create a client-server model in which all communication and data distribution are located in one host. This idea had to be rejected because of its inflexibility.

Performance and advanced functionality is another constraint. At least for now, high-end 3D visualization or video encoding is impractical in Java. This may be a transient situation. Digital signatures, Java performance improvements, and a growing body of advanced libraries written in Java may allow us to in the future to implement the entire system in this language. For the time being, we felt that the benefit of building a flexible, powerful system with rich functionality outweighs aestheticall architectural concerns.

Demon implementation

We use the LiveConnect technology introduced by Netscape Inc. in its Navigator version 3.0. This product provides means for Netscape components such as Java applets, JavaScript scripts and plug-ins to talk to each other. The *demon* has been implemented as a plug-in. Using LiveConnect mechanisms, each applet residing in the same page with the plug-in has access to the plug-in handle. Message passing between plug-in and the applet is achieved by calling appropriate methods of each other. In case of standalone applications, the connection to the demon is implemented with use of standard socket communication.

Plug-ins are usually written in C (the API provided by Netscape uses this language). To improve the porting procedure of the only platform dependent component of TANGO we associate Java code with the native C code and implement large parts of the plug-in code in Java. The only the part of the plug-in implemented is C is the set of routines used to start standalone applications and to provide intranode applet-application message passing. In this way we have minimized porting effort to an absolute minimum. TANGO is currently available on both Unix and Wintel platforms.

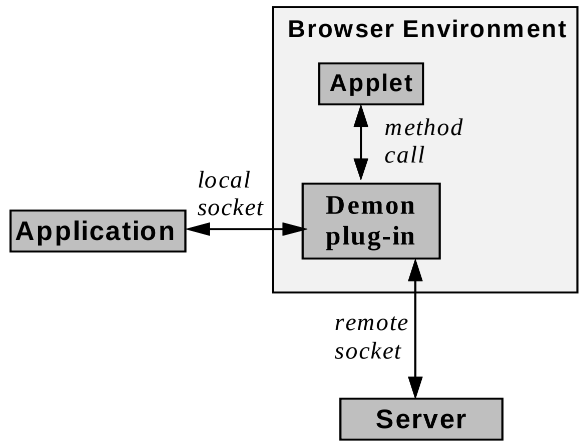


Figure 2. TANGO communication schema

Application API

Porting application to TANGO is facilitated by a simple API. Detailed API description is beyond the scope of this article. The basic functionality includes `register`, `write`, and `receive` as well as a set of functions used to control state of the applets. In case of Java we provide a package containing two classes: the `AppletBase` class implements functions mentioned above and the `Message` class provides a wrapper for user-defined messages. For C language the API is defined as a set of functions provided in a library. Message is implemented as a structure.

At present , messages are represented by byte arrays. It is programmer's responsibility to convert these data properly. Object serialization and RMI will be used as soon as these technology is available from JavaSoft.

Server implementation

The *central server* is a multithreaded standalone Java application. Server keeps all the information about connections and users in dynamic data structures and also stores them in the database. In our implementation we use JDBC package and Oracle RDBMS.

Videoteleconferencing

For scalability reasons, the real time multimedia streams are not sent via central server. Instead, a distributed architecture akin to the Insoft's (currently Netscape) OpenDVE has been implemented. This architecture supports multicast. Session control remains with the TANGO session manager. Audio/video decode/playback capability is implemented in Java for certain supported codecs. Backend database repository and playback capability for the audio/video streams is provided by a random access video server and is a part of the global system session archiving/retrieval facility.

References

Gosling, B. Joy, and G. Steele. (1996) The Java Language Specification. Addison-Wesley Developers Press, Sunsoft Java Series

Schooler E.M. (1996) Conferencing and Collaborative Computing. Multimedia Systems 4: 210-225

Grudin, J. (1990) Computer-supported cooperative work: history and focus. IEEE Computer 27: 19 - 26

NCSA Habanero Project (1996) http://www.ncsa.uiuc.edu/SDG/Software/Habanero/

Caltech Infospheres Project (1996) http://www.infospheres.caltech.edu/