TANGO - a Collaborative Environment for the World-Wide Web

Lukasz Beca, Gang Cheng, Geoffrey C. Fox, Tomasz Jurga, Konrad Olszewski, Marek Podgorny[1], Piotr Sokolowski, Tomasz Stachowiak, and Krzysztof Walczak[2]

Northeast Parallel Architectures Center, Syracuse University 111 College Place, Syracuse, New York 13244-4100, USA[3]

Abstract

Geographical and logical growth of the World-Wide Web is accompanied by a fast technological development. Web can be successfully used as a platform for implementation of diverse applications. Distributed and collaborative systems are among the most challenging Web applications. TANGO is an integration platform which enables implementation of Web-based collaborative environments. The system provides means for fast integration of Web- and non-Web-applications into one multi-user collaborative systems. In this paper we describe the functional model, requirements, system design and certain implementation issues of the TANGO system.

---

[1] Corresponding author: marek@npac.syr.edu

[2] On leave of absence from EFP, Poznan, Poland

[3] This research was supported by the U.S. Department of Defense under Rome Laboratory Collaboration & Interactive Visualization Contract No. F 3062-95-C-0273-01 and by Polish KBN grant No. PB1173/T11/95/09

1. Introduction

During the recent years we observe a still increasing role of computer networks and technologies related to them. Especially, a significant influence of the Internet is very well visible. Internet became not only a technological but also a psychological and sociological phenomenon.

Internet connects millions of computers distributed all over the world, introducing means for global computer communication. Web can now be perceived as one gigantic virtual machine, being able to serve information, computer power and all kinds of resources connected to it. Together with the geographical and logical growth of this network we observe a very fast development of the underlying technologies.

Many different methods of information exchange have been used for years, including file transport, electronic mail, and gopher, to mention a few. However it was the World-Wide Web technology that introduced a real revolution into computer world. Some of the reasons of Web popularity are rich information content, attractive and intuitive interface, and platform independence. Web allows presenting various types of data (text, images, links to other documents) together in a convenient hyper-text based form. The World-Wide Web gained quickly immense acceptance and over the years it has proven its stability and robustness.

At its beginnings, the World-Wide Web was little more than a simple way to publish and access electronic documents. However, as its popularity continued to increase, new technologies were being developed and introduced. These techniques provided users with new capabilities. First there were techniques of presenting more complex and attractive kinds of data such as sound and video. Then the World-Wide Web started to be perceived as a possible platform not only for information storage and exchange, but also for the computing in general sense [1]. Users want to migrate with their software into this new, attractive network environment. One of the natural trends is to start using the Web as a collaboratory environment. This paper describes a methodology to turn the Web into such an environment, supporting both synchronous and asynchronous collaboration modes.

The rest of this paper is organized as follows: Section 2 discusses rationale for the presented work. Section 3 presents functional model of the system. Section 4 compares our work with few other projects that attack theissue of a Web-based collaboratory. Section 5 provides the reader with description of system architecture and implementation. Finally, section 6 contains our concluding remarks and further work reference.

2. 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 Web success of the Web, but it also strongly limited its further development. The basic WWW model does not support any real interactions. The introduction of the CGI (Common Gateway Interface) mechanism enriched and extended the Web but it did not solve the interactivity problem. At this stage of the World-Wide Web development, it was still very difficult to create efficient and scaleable tools for cooperative work on the web. The situation has changed with the introduction of several new technologies.

In 1995 Sun Microsystems introduced Java [2]. 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 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 [3] which used extended Postscript as both a programming language and a 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 for Web applications 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 [4]). Despite of the extensive body of research, collaboratory systems are still hardly out of the experimental stage. 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 rather 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 InPerson, 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 [5].

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 backend. 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. a useful distributed collaborative systems can be built by integration of slightly modified standalone applications together with some user and session management mechanisms

3. System's functional model

Before we turn to a technical description of the TANGO system, it is important to discuss its basic functionality. We perceive TANGO as a modest prototype of an open, extensible system that provides a technological framework for building subsequent generations of collaborative systems. It is therefore useful to make a distinction between the functional model and actual implementation. We tried to build a functional model that addresses and embraces a number of unresolved issues in the field of Computer Supported Collaborative Work (CSCW). We hope this model to be sufficiently generic to serve as a framework for ongoing implementation process. Implementation methodology is often a compromise dictated by the current state of the art of the basic Web technology we are building upon: web browsers and the Java Virtual Machine. This state of the art is often a state of misery, as it should be expected from infant technology. However, true to our belief that an efficient collaboratory can only be built by an arduous process of getting feedback from the users, we decided to provide a working implementation of the system regardless of the evolving technology. We are confident that the new technology trends and tools will help us enrich and better the system as we incorporate them into future implementations of the basic functionality.

The functionality that the system design addresses a number of the following issues, vital to the further development of collaboratory systems:

- *TANGO is aimed at creation of shareable information spaces:* The basic paradigm of many traditional collaboratory systems is to connect identical instances of few applications and allow them to exchange data objects or streams. We consider this paradigm way too restrictive. There is no basic reason to restrict collaborating people and teams to look at the same information in the same way or using the same tools. Collaboratory system should support *coordinated but independent views of related information*. Different presentation of information may be needed for different reasons: a 2D GIS display on the command and control main screen may be sufficient while a mission planner may want to see a much higher resolution 3D representation of the same information. In other cases, different information views my be displayed on workstation with different performance or graphics capabilities. In TANGO, we allow different applications to exchange messages and act accordingly. The applications exchanging information can reside on either the same node or on different machines. This is an important feature, since, independently of supporting much richer collaboration model, it allows us to build complex applications from small, reusable modules. Consider, as an example, a tandem of a chat and a web browser. This tandem is replicated on multiple nodes. As the users chat, they may pass information about interesting URLs in the chat window. In TANGO system, this information will be automatically recognized and passed to the set of browsers working in the collaborative mode. Another example is a simulation system created from a customized TANGO environment: a simulation engine sends messages to both local and remote applications of various kinds, creating a war game or a crisis management center training module. In both examples the users of the system have full access to the enormous *information resources of the entire Web*. This information may be presented to them either via a standard, familiar web browser interface, or via a set of specialized filters that tailor information display to a specific task.

- *TANGO is very tightly integrated with the Web:* This statement applies both to TANGO as a functional model and to its implementation. In the functional sense, tight integration with Web implies access to the entire informational potential of the Web. This is critical for the construction of the shareable information space discussed above. Further, designing TANGO functionality we have opted for the *self-distributing software model* for collaboratory applications. While this decision created a number of difficult software implementation issues, it also makes the system very easy to install, use, and extend. Implementation issues will be discussed later, here we wish to point out that TANGO is the first collaboratory system tapping full potential of applet download capability provided by Java.

- *TANGO provides support for all synchronous collaboratory functions:* Built in a stateless Web environment, TANGO is a statefull system. Basic design supports all functions of a synchronous collaboratory system, including session management, data/event distribution, flexible floor control, and multiple, configurable security levels. The system does not impose any restrictions on a number of concurrent sessions, users, or collaborative applications.

- *Asynchronous collaboration is supported via database back-end:* Session record and playback capability has been designed into the system as its fundamental component. This capability applies to both events and data streams and can be used review collaborative sessions in asynchronous fashion. Playback/review capability is provided also for real-time continuous data streams such as audio and video.

- *Distributed visualization and manipulation of multimedia information streams*: TANGO provides scaleable multimedia support. Recognizing different performance requirements of continuous multimedia streams, TANGO provides mechanisms for decentralized distribution of the such streams under TANGO session control.

- *Language independence:* While almost entire TANGO runtime and most of the applications are written in Java, there is no restriction on the language used to implement collaborative applications compatible with TANGO. This design decision appears to violate the aforementioned requirement of application downloadability. Actually, methodologies for automatic distribution of applications written in languages other than Java exist and we intend to incorporate them into our implementation. We have decided however that, at least at present, restricting ourselves to Java would prevent us from extending our systems to such important domains as 3D visualization, high quality video streaming, and videoconferencing (especially multimedia encoding). In addition, TANGO provides a truly unique support for applications written in JavaScript. Our ability to do so stems from the earlier decision of very tight integration of TANGO with Web.

- *Extensibility:* As stated above, we firmly believe that ultimate success or failure of a collaboratory system depends on its application set. Video, audio, and whiteboard conferencing is just a seed of a collaboratory. Domain specific applications must be either implemented from scratch or ported from their stand-alone versions for a collaboratory to become truly useful. For this reason, TANGO APIs for Java, C/C++ and Javascript have been written and well documented.

4. Related work

TANGO is not the only web-based collaboratory system. There are multiple ongoing projects in the area of Web based collaboratory systems. The projects we are aware of represent a rather interesting variety of approaches in aspects such as focus, design methodology, and technological sophistication. A complete review of these projects is well beyond the scope of this paper, but we would like to be able to locate TANGO in the landscape of the ongoing activities.

Many of the collaboratory projects are clearly ad-hoc creations. Examples of such approaches are Stanford NRE Collaboratory System [6] and the CORE system from Environmental Molecular Sciences Laboratory (EMSL) at the Pacific Northwest National Laboratory [7]. Built by scientists to support their own work, the systems take shortcuts to build a minimal practically acceptable collaboratory functionality. Technically, both NRE CS and CORE rely on CGI/MIME technology to start helper applications (either a set of standard tools (NRS CS) or a custom application (CORE)) on the collaborators' desktops. Focus of these systems is to provide a quick solution for CSCW demands of the distributed research projects that cannot be satisfied by currently available commercial products. These systems exemplify first generations of the Web based collaboratory, they are already technologically obsolete, and, in absence of a scaleable system design, they probably cannot adopt new technological developments. However, both these attempts provide testimony to the necessity of building efficient CSCW systems. TANGO entirely subsumes functionality of such systems.

Another approach to Web collaboratory is presented by a project of Center for Information Systems Management at the University of Texas [8]. Technologically rather simple, this project is built around the informational contents of the collaboratory. Many ideas of this project are quite generic and probably can be adopted by any advanced collaboratory system with focus on providing shared access to domain specific information repositories. Our system does not address issues of this type.

We are also aware about three collaboratory systems written in Java. All of them have certain elements in common with TANGO. The Java Collaborator Toolset (hereafter referred to as JCT) of Old Dominion University [9] uses applets (at least according to the documentation; there is no operational demonstration available over Internet) as collaboratory modules. To provide event and data sharing, the AWT of the JDK is replaced by a custom collaboratory toolkit. Obvious advantage of such a system is simplicity of application porting. However, we consider this approach to be too restrictive: JCT is shared X written in Java. The only possible form of collaboration is via identical instances of application. We believe that TANGO represents much more general and flexible approach without sacrificing any of the advantages of the JCT.

Two other systems are NCSA Habanero [10] and Caltech's Infospheres [11]. Habanero is a project of great potential. The main differences as compared to TANGO are: (a) collaboratory modules are Java applications, not applets, so Habanero does not support software downloadability and, hence, is not really a Web based collaboratory; (b) for now, Habanero is limited to Java applications (although there is no fundamental reason for this restriction); © Habanero does not offer asynchronous collaboration since there is not database support (again, there could be). We do not know why Habanero designers have chosen applications over applets implementation, buy we certainly can see advantages of such approach. Habanero architecture is not dependent on ill-defined and ever changing behavior of Web browsers and does not suffer from performance limitations imposed by the browsers. The price paid for this convenience is a less powerful system: Habanero's affinity to Web and its ability to tap Web's enormous informational resources do not appear to match TANGO's

As for the Infospheres [11], the focus of this project seems to be different. In their context, collaboratory is just an example of a distributed system. Infospheres project has a very strong theoretical component [12] investigating compositional systems supporting peer-to-peer communications among persistent, multithreaded, distributed objects. We currently study Infospheres project to investigate how to use these ideas to improve TANGO implementation. Our understanding is that the current implementations of the Infospheres system is functionally more similar to Habanero than to TANGO.

5. TANGO: Architecture and Implementation

5.1. System requirements

We have based design and implementation of the TANGO system on the following formal 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 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 either locally or remotely

5.2 Low-level collaboratory functionality

The main functionality provided by the system consists of the following elements: (a) session management, (b) communication between collaborating applications, © user authentication and authorization (d) event logging.

5.2.1. Session management

A *session* is a group of application instances currently working together in the 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.

In all 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.

There are three possible actions which may change state of a session:

Creating a session: Session is created when a participant launches a local application. On the beginning, the session contains only this one application instance. The participant becomes automatically the master of this session. Other participants may then join this session.

Joining an existing session: There are two possible ways of joining an existing session:

- by launching a new application and connecting it to a session, or

- by connecting to a session with already existing application instance - in this case the application performs "*Leaving a session*" scenario and then connects to another session;

Leaving a session: When a participant leaves a session, there are two possible cases which need consideration:

- participant *is not the master* of the session - in this case leaving the session means only removing given application instance from the session,

- participant *is the master* of the session - in this case the session ends. All participants are removed from the session, the application ends, and the session is deleted.

TANGO does not restrict the number of concurrent sessions. There may be multiple independent sessions of applications of the same type. If messages from one application are compatible with application of another type and the compatible application on the TANGO node on which the first application is running, the messages to compatible applications will be distributed transparently.

5.2.2. Communication

Only applications belonging to the same session can communicate. System provides means for this communication. It is implemented on the basis of message passing. If an application sends a message, this message will be delivered to all other compatible applications in this particular session.

There are two major types of messages: (a) control messages and (b) application messages.

Control messages are generated by the system for communication between the server, daemons, and control applications. These messages serve functions such as logging users into the system, establishing sessions, launching applications, etc. Control messages are invisible for user applications.

Application messages are main means of communication between 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. The application specific communication protocol is always defined by the application itself using TANGO API.

At present, TANGO messages are sent as strings. A preliminary implementation using object serialization and remote method invocation exists and will be moved into the working implementation as soon as Java version 1.1 is supported by Web browsers.

5.2.3. User authentication and authorization

System provides means of user authentication and authorization. All user data is kept in a database and is used by the main server which acts as a router of all control and application messages. If a user fails authentication s/he cannot open sessions and connect to the system. It is possible to associate certain application functionality with certain level of privileges. If the user does not have appropriate permissions, the server will refuse accomplishing requested action. User data can only be accessed by special application via custom system administration application. System security measures can be deactivated using TANGO server configuration tools.

5.2.4. 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 asynchronously reviewed.

During replay of the system activity the database acts as a source of information. Since it keeps application messages as well as control messages, user status, sessions configuration and applications running can be restored. Stored application messages enable restoring of activity inside applications.

Event logging functionality is a critical system elements designed to support *asynchronous collaboration modalities*.

5.3. System architecture

5.3.1. System elements

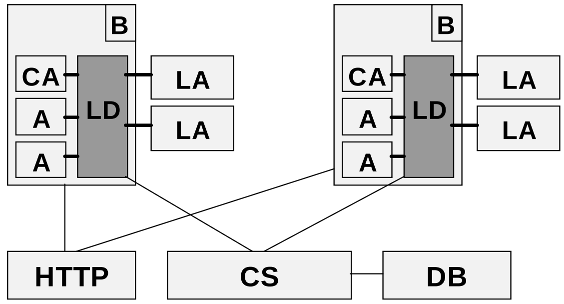
In Figure 1 the global architecture of the system is presented. The system consists of the following components:

Local daemon's main tasks are (a) maintaining two way communication between user applications, applets and central server, (b) launching local applications, © passing messages between applications running on the same node, and (d) providing certain system level functionality not normally available to Java applets. TANGO daemon is implemented as a plug-in to Web browsers. The daemon is the only operating system dependent core part of TANGO. At present, implementations exist for Netscape v. 3.0 on several platforms. For Microsoft's Internet Explorer the daemon is implemented as an ActiveX control.

Central server is the main communication element. All local daemons communicate with the central collaboratory server. Collaboratory server maintains the system state data (participants, applications, sessions, etc.). Its main tasks are keeping the state of the entire system, routing of messages between applications participating in each session, and recording all events in the back-end database. Currently, TANGO system is restricted to only one collaboratory server. We are investigating a model for inter-server communication. This sub-project is directed towards extended functionality of a collaboratory systems such as automatic discovery of potential collaborators.

Local Applications: All user applications which run as standalone programs we call *local applications*. Local application may be written in any programming language. All local applications communicate with the local daemon by the use of sockets. The daemon 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 daemons. Java applets communicate with local daemon by calling its method functions.



CA - Control Application
A - Java Applet
LA - Local Application
LD - Local Daemon
CS - Central Server
HTTP - HTTP Server
DB - Database
B - Web Browser

Figure 1. Global architecture of the TANGO system

Control application is a specialized application which acts as an interface to a user [13]. Control application is used to launch applications locally or remotely, to create and connect existing sessions, to 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 these features only he/she is allowed to use.

Control application communicates with the system by the use of the local daemon. The communication between control application and local daemon is different than in the case of standard Java applets, because control application can also generate system messages.

5.3.2. Event flow

TANGO transparently supports event distribution. However, in contrast to the approach taken in [9], the application developer is free to decide which application events need to be distributed. This decisions are implemented by inserting TANGO API calls into applications. How and where events are distributed depends on the session management layer.

Special sequences of events are invoked by the instances of Control Applications in response to the following user actions:

- Entering the system
- Leaving the system
- Launching local application
- Leaving the session
- Joining a session with new application

- Joining a session with existing application
- Launching remote application
- Switching to master mode (or, more generally, floor control events)

For detailed description of the event flow we refer the reader to TANGO documentation.

Another special case of the event flow is invoked when an application joins an existing session. For certain applications it is important to initialize the new instance with the current state of existing instances (as an example, consider a shared drawing program). For applications that require complex initializations TANGO supports a special protocol extension. We expect this particular aspect of TANGO implementation to change after object serialization/Remote Method Invocation of Java 1.1 become available in the browsers.

5.3.3. TANGOsim extension

An example of TANGO architecture flexibility and extensibility is TANGOsim - an discrete event simulator [14] . 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”.

5.3.4. Videoteleconferencing

For scaleability 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[4] [15]. 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. Currently, we support two audio (GSM and ADPCM) and three video codecs: very low bit rate H.263 codec (phone line bitrate), medium bit rate H.261 (ISDN line bit rate), and the YUV9 format (LAN bit rate). Back-end database repository and playback capability for the audio/video streams is provided by a random access video server [16] and is a part of the global system session archiving/retrieval facility.

5.4 Implementation Issues

One of the most important goals while designing TANGO was its portability. The World-Wide Web is composed of a plethora of systems. Creation of software versions for each platform may be a significant effort. Java language provides means to overcome this difficulty. By using one uniform environment on each platform it resides, Java allows to run the same code everywhere.

---

[4] Our original intent was to use the announced Netscape LiveMedia architecture. LiveMedia was supposed to be a re-engineered OpenDVE from Insoft, but it never materialized except as CoolTalk, Netscape audio conferencing tool . The original Insoft OpenDVE had insufficient performance, undocumented interfaces, and a fair share of bugs. We have re-implemented entire functionality of OpenDVE from scratch. This implementation is available as NPAC Conferencing System [15].

Ideally, we would like to implement all system components using only Java. This approach turned out, however, to be not feasible for several reasons. The most important are security constraints imposed on Java. If we wanted to have our clients in form of Java applets, we could not force them to connect to the *daemon* residing on a different host. 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.

5.4.1. Daemon implementation

Daemon provides a mechanism for Netscape components such as Java applets, JavaScript scripts and plug-ins to talk to each other. The *daemon* has been implemented as a plug-in. Using LiveConnect mechanisms [17], each applet residing in the same page with the plug-in may obtain its handle. Message passing between plug-in and an applet is achieved by calling appropriate methods of each other. In case of the local applications, the connection to the daemon is implemented with use of standard socket communication. Comparison of standalone applications and applets is presented in Table 1. In Figure 2 the communication schema is presented.

activity	**applet**	**standalone application**
starting new program	dynamic HTML generation	system call (platform specific)
communication with daemon	mutual method calling	local socket connection

Table 1 Applets vs. applications - comparison

Although plug-ins are usually written in C (the API provided by Netscape uses this language), it is possible to associate Java code with the native C code and in this way implement some parts of the code in Java. Plug-in written in Java is not treated in the same way as an applet. In TANGO, ~90% of the plug-in is written in Java. Only the part used to start standalone applications is implemented in C. Using such approach we obtain means to communicate between various applications on the client. The plug-ins must be created separately for each platform, but since the system-dependent “C part” is very small, this is a relatively easy task. At the time of this writing (February ’97) TANGO plug-ins are available for Solaris 2.4+, Irix (5.3, 6.2, 6.3), Windows ’95 and Windows NT. Linux version is expected soon.

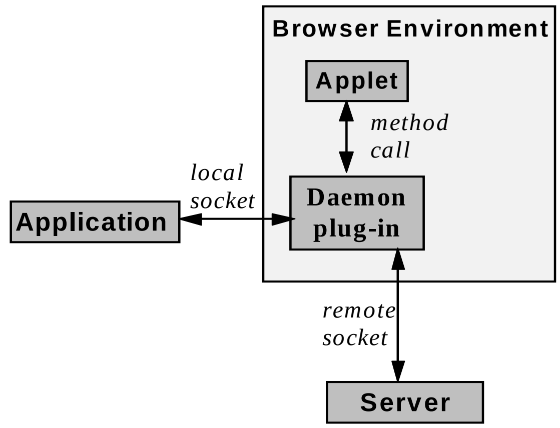


Figure 2. TANGO communication schema

5.4.2. Server implementation

The *central server* was implemented as a standalone Java application. Its main part listens for incoming connections on a known port. For each new connection a separate thread is started. 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 database system as a back-end.

5.4.3. Control application

Control application [13] has been implemented as a Java applet. A small part of it is launched on the beginning giving a user a possibility to login to the system. Then the rest is downloaded from the HTTP server later, after user authentication, and depends on the user privileges.

5.4.4. Application API

Tango API [18] is used to port a standalone application into a shared application that can be collaboratively run under the Tango system. Using the Tango API, all the message passing and event routing are completely transparent to the developers. The developers only need to define when and what the application will share with all participants of the same application. This is usually done by inserting Tango API into the application code, without modifying any line of the existing code.

Tango API has two major categories:

1. *Tango Core API* is used for an application to communicate with other participants of the same application under the Tango collaborative infrastructure. It is assumed that the same application will act both as a sender (producer) and as receiver(s) (consumer) of messages/events when multiple copies of the same application run on distributed (different) host machines over the Tango system. Therefore, the main purpose of the Tango Core API is to allow an application to register into the Tango runtime system, and for events to be shared among participants to send and receive them over the Tango system, without knowing how an event is distributed over the network. More specifically, when porting a Tango application, a developer must define:

    - When to share an event. Usually this occurs when the event handler of the application wants to share a significant change on the GUI of the application, for example, a button is pressed.

- What to share. Tango system is developed to be a light-weight event-driven system that can run over a wide area network. It is assumed that only events or application-defined protocol data (not bulk data) need to be sent that will be used to drive the application behavior. In the current implementation, Tango Core API provides message objects to convert/pack any event into a binary format before/after it can be sent/received. In the near future, we plan to use Java RMI/Object serialization mechanism that will be supported in the version 1.1 of Java JDK from SUN to simplify the current message API.

- What to do after an event is received. Because a Tango application usually runs in the single program multiple data (SPMD) mode, it is the application's job to define the program behavior after an event is received. Usually, the behavior has been defined in the event handler part of the application when the event is handled by the originator.

However, when porting a Tango application, a developer *does not need to define/know*:

- What are the Tango runtime components and where are they running ?

- How to login/connect to the Tango runtime ?

- How to start/join a tango application/session ?

- How an message is being distributed ?

- Where and how the sessions and users information are kept in the system ?

2. *Tango CA API* allows an application to access the current runtime information of the Tango system, such as sessions, participants, modes, identifications, application lists etc. This is usually used for the applications that themselves are collaborative in nature, such as chat, whiteboard, audio conferencing etc.

Currently, we provide two Tango APIs for applications written in different languages, including Tango Java/Applet API, Tango C/C++ API, and Tango JavaScript API.

JavaScript API is a rather interesting case. Using it, we are able to build client-side collaborative applications. TANGO-ized JavaScript applications can be synchronized within a "collaborative browser" based on a standard browser of any vendor. Further, by combining TANGO JavaScript API with HTML pages preprocessing using technologies found in Web spiders we are actually able to automatically turn *each* Web page into a collaborative entity.

6. Conclusions

We have designed, implemented, and distributed TANGO, a web based collaborative system. The system extends Web paradigm to true collaborative computing and well beyond the chat and shared whiteboard concept. TANGO framework is open and extensible. In it minimal version the system fully supports self-distributing software model, due to being written almost entirely in Java and completely integrated with web browser environment. In the extended version, TANGO supports collaborative applications written in any language, including JavaScript, and provides a simple and fast methodology to turn the existing applications into collaborative modules. The system supports both synchronous and asynchronous collaboration modes. TANGO is also very portable: in the short life of the project, versions running on a number of UNIX and Wintel platforms have been implemented.

Our future work has two focal points. One is improvement and further development of the core TANGO runtime and APIs. Here, the project is driven by the technological advances of commercial Web tools and by academic research in distributed Internet systems, such as Caltech's Infospheres [12]. Another is a rapid and vigorous development of the domain specific application modules and deployment of the system in the domain specific testbeds. At present, TANGO supports some two dozen of collaboratory modules, from simple chat and whiteboard applets, via advanced desktop videoteleconferencing, to complex, domain specific applications as a command and control center prototype, weather analysis and prediction system, 2D and 3D GIS visualization, or a Virtual University distance learning application. We expect to extend this list by both enriching the current application domains by adding new modules and by entering new domains, like telemedicine.

Acknowledgments

One of us (Krzysztof Walczak) acknowledges financial support of NPAC during his stays in Syracuse. We also appreciate help we have received from the NPAC VoD and GIS teams: Kemal Ispirli, Greg Lewandowski, Pawel Roman, Remek Trzaska, and Bart Winnowicz.

References

[1] Fox, G.C, Furmanski, W, "Java and Web Technologies for Simulation and Modeling on Computational Science and Engineering", Proc. of the 8th SIAM Conf. On Parallel Processing, Minneapolis, March 1997.

[2] Gosling, B. Joy, and G. Steele. "The Java Language Specification." Addison-Wesley Developers Press, Sunsoft Java Series, 1996

[3] Gosling,J., Rosenthal, D.S.H., Arden, M., "The NeWS Book", Springer-Verlag New York 1989

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

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

[6] Stanford NRE Collaboratory System, http://laputa.stanford.edu/collab/collab.html.

[7] EMSL Collaborative Research Environment (CORE), http://www.emsl.pnl.gov:2080/docs/tour/index.html

[8] "Collaboratory in Cyberspace", an on-line paper by Anitesh Barua, Ramnath Chellappa, and Andrew B. Whinston, http://cism.bus.utexas.edu/ram/papers/joc/joc.htm.

[9] Java Collaborator Toolset, http://www.cs.odu.edu/~kvande/Projects/Collaborator/

[10] NCSA Habanero Project, http://www.ncsa.uiuc.edu/SDG/Software/Habanero/

[11] Caltech Infospheres Project, http://www.infospheres.caltech.edu/

[12] K.M. Chandi, A. Rifkin, "Systematic Composition of Objects in Distributed Internet Applications", Proc. of. The 30th Hawai International Conference on System Sciences, January '97, Maui, Hawai

[13], Jurga, T., "Control Application for Java/WWW-base Collaboratory Environment of Tango System", Internship Report, NPAC, Syracuse, January 1997.

[14] Beca, L., Cheng, G., Fox, G.C, Jurga, T., Olszewski, K., Podgorny, M., Walczak, K., "Web Technologies for Collaborative Visualization and Simulation", Proc. of the 8th SIAM Conf. On Parallel Processing, Minneapolis, March 1997.

[15] NPAC Conferencing System, http://trurl.npac.syr.edu/toms/Projects/NCS/ncs-info.html; Stachowiak, T., "NPAC Conferencing System", Internship Report, NPAC, Syracuse, January 1997.

[16] NPAC Video on Demand project, http://trurl.npac.syr.edu/rlvod

[17] The LiveConnect / Plug-in Developer's Guide, http://home.netscape.com/eng/mozilla/3.0/handbook/plugins/index.html

[18] "Tango API for Developers", http://www.npac.syr.edu/tango/TangoAPI.html