Secure Group Browsing

R Jain, N Jain
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
India - 781039

S Nandi
School of Computer Engineering
Nanyang Technological University
Singapore - 639798
E-mail: assukumar@ntu.edu.sg

Abstract— **This paper presents a model of a system that enhances the current features of the Net by providing the users a facility to securely browse the Internet together. It thus provides the *'feeling of togetherness'* on the Internet. The proposed system is generic and thus can be used to *'group enable'* any existing and running site. It is a module that can be plugged onto existing systems without affecting their normal behavior. It would thus make concepts such as group shopping, group Medicare centers, group e-study centers etc., on the Internet a reality.**

I. Introduction

The horizon of Internet is broadening everyday. In its effort to simulate the existing activities on the Net, be it banking, or stock sales, shopping or bill-payments, their counter parts are present on the Internet in the form of On-line Banking [10], On-line Stock Market [11], e-shopping [11], Electronic Bill Payment Systems [4–9], On-line Schools, On-line Medicare Centers and similar applications add new dimensions to the Net. E-mails and chat provide a medium to share our views and communicate with others. Facilities like audio or video conferencing brings people face to face thereby overthrowing the geographical distances.

With the advent of more and more such applications we are heading towards the concept of Internet Life wherein most of our needs would be satisfied on the Net. Apart from emulating the current systems, one of the greatest strengths of the Internet lies in its inherent nature of being wide spread and within reach of almost all parts of the globe. Group activities, connecting people from all over the globe, thus becomes a logical possibility. This concept of *'group activities and togetherness'* on the Net has not been explored completely.

Presently, group activities on the Internet are limited to audio/video conferencing [2] and various chatting sites, but that limits the scope of interaction only through audio/visual means or by exchanging ideas through chatting.

If a group of people in different geographically locations, want to take part in various activities like shopping or education, in a group, then they are left with no solution. An ad hoc solution to this could be that the entire activity is performed as isolated users (i.e. *www.fabmart.com*) and then communicate with each other via chat or phone while browsing the site. This solution however, is not tailored to the problem and furthermore, does not use the capabilities of the Internet to its apogee.

This work has been carried out at Indian Institute of Technology Guwahati, Guwahati -781039, India.

To make group activities possible on the net, using existing infrastructure, we propose in this paper a new horizon in-group activities on the net viz. *'Secure Group Browsing'*. This concept enhances the capability of the existing systems by making them accessible by groups of users. It ameliorates the systems without modifying them and tampering with their integrity. This paper contains the details of a system, which can *group enable* already existing sites. Thus if a site provides applications/facilities like e-shopping, e-banking, information sites etc., by making use of the proposed system users would be able to access all these facilities in a group or otherwise. The proposed system could hence make possible facilities such as, *'group e-shopping'*, *'group e-education'* or *'group e-banking'*. The system takes care of various issues of group formation, group management, authentication, and transmission of response to all group members and security of the transactions.

This paper is organized as follows - Section II describes the system from a users point of view. Section III discusses various design constraints to be considered. Section IV describes in details the architecture of the proposed system. This is followed by a detailed discussion on the working of the system in section V. Section VI deals with various security aspects of the proposed system. Section VII contains an analysis of the performance of the said system.

II. User Interface

To make it possible for all members of a group to see what the others are viewing and to give them the feel of being in a group;
1. Each member should be able to view what the other members are viewing.
2. Each of them should also be able to browse independently.
3. There should be a communication channel through which group members can interact with each other. This would enable them to share their comments, advises and suggestions about the various things they view.

To provide the users a feeling of togetherness the *'User Interface'* of the proposed system takes care of the issues discussed above. Each member has a set of windows on his screen. One of these windows is the member's own window, in which he can browse the site, thus called the *'browsable area'*. The other windows belong to the other group members and can only by viewed, thus called the *'read-only'* area. Each member can view what the other members are viewing in the read-only windows corresponding to those members. To provide the communication channel between the group members the system provides the facility to chat.

III. Design Constraints

With every new application, the Internet is progressively becoming an aggregation of *heterogeneous* systems. Different systems/applications employ different technologies and have to deal with different issues. For instance the requirements of on-line education systems (like *www.egurucool.com*) are different from those of an on-line shopping mall (like *www.amazon.com*). To make the proposed system easily deployable on the net and make it free from technologies lock so that it can be easily plugged onto existing sites and applications the following issues have been considered.

A. Use of existing technologies

Since new technologies and applications based on these technologies are on a rise, issues of backward compatibility and inter product compatibility are becoming more and more prominent. In order to work with minimum or no changes to the existing Internet infrastructure, the system makes use of the existing/prevailing technologies. For example most of the transaction on the Internet are done on HTTP [3]. Routers/servers/proxies have certain algorithms/protocols customized for HTTP to improve the performance and for the applications to function correctly. The proposed system thus uses existing Internet technologies.

B. Add-on Module in the existing applications

The system is implemented as an add-on module so that it can be plugged on to existing systems without requiring any change in their current working. To do so otherwise would require customized solutions for the different sites to enable group browsing.

For the proposed system to be an add-on module, it is designed to be transparent to the site/application. It acts as a gateway, which intercepts the requests/responses being sent to and from the site, and makes changes to them without affecting the normal working of the site. The system ensures that group member's browse in a group, but as far as the site is concerned it would be unaware of the existence of the system and would function normally.

C. No Changes on the user side

From the end-user's point of view there is no change in the way he browses the various sites. He will still be able to browse the sites in a group or otherwise using the standard browsers he uses. If the system required a customized application to group-browse the site, it may not have been acceptable to the users. More so, these popular browsers keep updating with new features. Keeping the application in tune with these developments would require a considerable effort.

IV. Architecture of the proposed system

As per the design considerations mentioned in the previous section the architecture of the proposed system is as shown in Figure 1. It consists of a **Group Management System (GMS)**, with its set of databases (GMS Databases in the figure). This add-on module sits before the application server and intercepts the requests, before they are forwarded to the application server and the responses, before they are sent to the client.

In the proposed design, GMS handles the *Group Formation and Maintenance.* The requests made by the members in the group are intercepted by it. It passes only the requests made from the browsing areas of all the members to the application. GMS then acts as a coordinator whose responsibility is to send the information pertaining to the request made by a member to all other group members. It broadcasts the response given to this member to the other group members, so that the other members can also see the responses.

Figure 1 shows two persons browsing in a group and a third client browsing the site individually. As far as the application server is concerned, the requests it receives are as if they are coming from three different clients. It is the GMS that make it possible for the group members to view each other's area.

A. Roles of GMS

Based on functionality, the role of the GMS is categorized as follows.
(a) Authentication of the client: Authentication of the client is necessary to provide the users an assurance that the other persons in the group are actually the ones, they claim to be. If such an authentication scheme were absent then malicious users would be able to join the groups and do undesirable things, thus making an as such secure site prone to attacks. The GMS can use the available authentication schemes [16–21] or a variant of them.
(b) Group Management: The users who visit the site should be able to start a new group, join an existing group or leave a group. The number of groups and number of clients per group would change dynamically. These issues of group management are taken care of by the GMS.
(c) Broadcast the response to all members of the group: The response to any client request is sent to all the members of the same group only. The GMS broadcasts this in a time efficient and bandwidth efficient manner.
(d) Database Management: For the purpose of its functioning, auxiliary databases are required as shown in Figure 1. Issues of concurrent transactions, consistency and integrity of the database become critical to the design of the system [23]. Due to the frequent access of the database, the performance of the system depends to a great deal, on its design.
(e) Security Handling: GMS takes care of various security issues. The group members should be sure of the authenticity of the data they view as also of the other group members. Adding the proposed system to existing sites should not create a security hole into an as such-secure site.

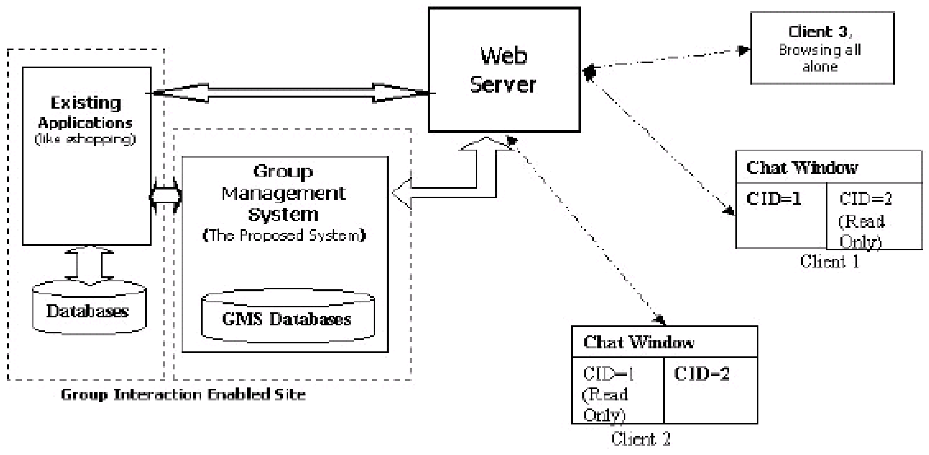
Web Server
Client 3, Browsing all alone
Existing Applications (like shopping)
Group Management System (The Proposed System)
Databases
GMS Databases
Group Interaction Enabled Site
Chat Window
CID=1
CID=2 (Read Only)
Client 1
Chat Window
CID=1 (Read Only)
CID=2
Client 2

Fig. 1. Basic Architecture (GMS as an add-on gateway module

B. Interaction between GMS and client browser

Figure 2 shows the client side structure of the system. It contains an agent program known as the **Actual Client Agent (ACG)**. Each user who is a member of group has an ACG. GMS sends the ACG to the user when he joins or starts the group (e.g. as a Java applet with the home page). ACG establishes a two-way communication channel with the GMS (a channel like a socket connection). GMS uses this channel to pass instructions and messages to the ACG, which in turn executes those instructions with the help of the browser. Corresponding to each ACG on the client side there exists a **Virtual Client Agent (VCG)** on the GMS, which is responsible for talking with the ACG on behalf of the GMS.

ACG apart from handling the message communication between VCG and itself also manages the user interface on the client side. It interacts with the client browser and asks it to fetch the recent responses of the other group members, as informed by the corresponding VCG. It also maintains the number of windows dynamically synchronized as users join or leave the group. It can manipulate the client side browser to produce the desired outcome.

When a member makes a request from his browsing area, GMS updates its database and forwards the request to the application. The application, unaware of the existence of the GMS, assumes that the request came from the client, processes it and sends the response, as it normally would. GMS intercepts the response and updates its databases. It sends a customized response to the client and saves a modified copy of the document with itself for the other group members. To broadcast the news of the new document to all the other members of the group it sends a message to the ACGs via their corresponding VCGs. ACG in turn take charge and gets the saved documents with the GMS (refer section V-B) corresponding to its group.

As mentioned above, the GMS make changes before sending the response to the client. These changes are made to URLs that will be used for group activities (refer section IV-D).

C. Anatomy of GMS

Figure 3 shows the basic modules in the GMS. It consists of three basic modules namely the HTTP Request Handler (HRH), the Garbage Collection Daemon (GCD) and the Virtual Client Daemon (VCD). Besides the above modules, GMS has a set of databases it uses. The functionality of each module is explained next.

(a) HTTP Request Handler (HRH): HRH is the primary module of the GMS. It sits between the web server and the application server without them being aware of its presence. It processes both the requests and the responses from and to the client, thus enabling group browsing.

(b) Virtual Client Daemon (VCD): When the ACG activates, it contacts the Virtual Client Daemon (VCD), which in turn creates a new VCG in the GMS corresponding to the ACG. The newly created VCG and the corresponding ACG have an established communication channel throughout the session. In the proposed system the communication channel is a socket connection.

(c) Databases: The GMS maintains two databases. The *Group Information Database (db1)* contains information pertaining to the groups. It contains one entry per group in the system. Each entry keeps the information pertaining to the group and a collection of Client Records. The *Stored File Information Database (db2)* contains information regarding the various files stored in the local storage of the GMS. It has an entry for each group. Each group record contains the entries corresponding to the stored files, which were browsed by the group members. One of the important information maintained by db2 includes the send bits. The send bits refer to the files, which have been received by each member (Refer to appendix A for more details).

(d) Garbage Collection Daemon (GCD): The Garbage Collection Daemon periodically cleans up and reclaims the local storage space. It also removes the unnecessary entries from the database. During cleaning up it primarily reclaims the space used by the local files, which have been served to all the members of the group (i.e. all send bits of a file record in db2 are marked).

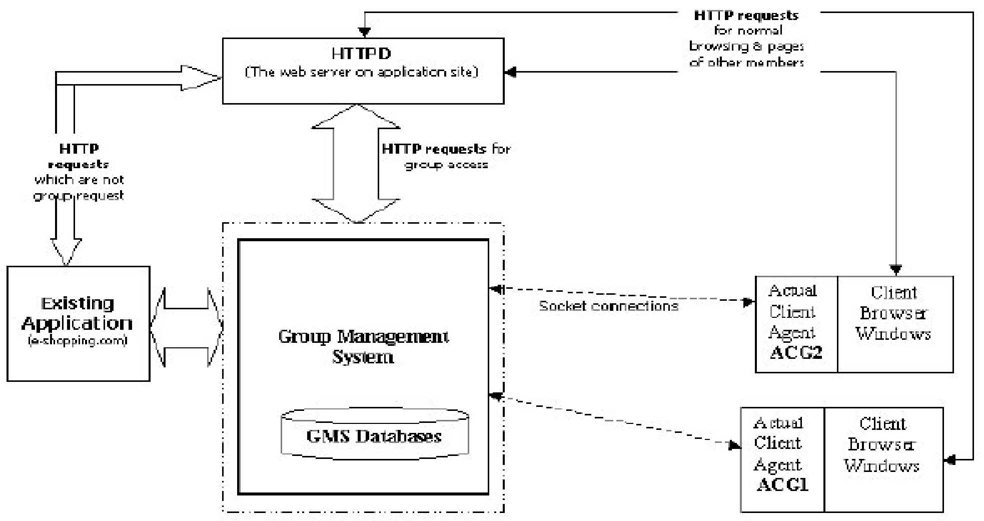
HTTP requests for normal browsing & pages of other members
HTTPD (The web server on application site)
HTTP requests which are not group request
HTTP requests for group access
Existing Application (e-shopping.com)
Group Management System
GMS Databases
Socket connections
Actual Client Agent ACG2
Client Browser Windows
Actual Client Agent ACG1
Client Browser Windows

Fig. 2. Client Side Agents (ACGs)

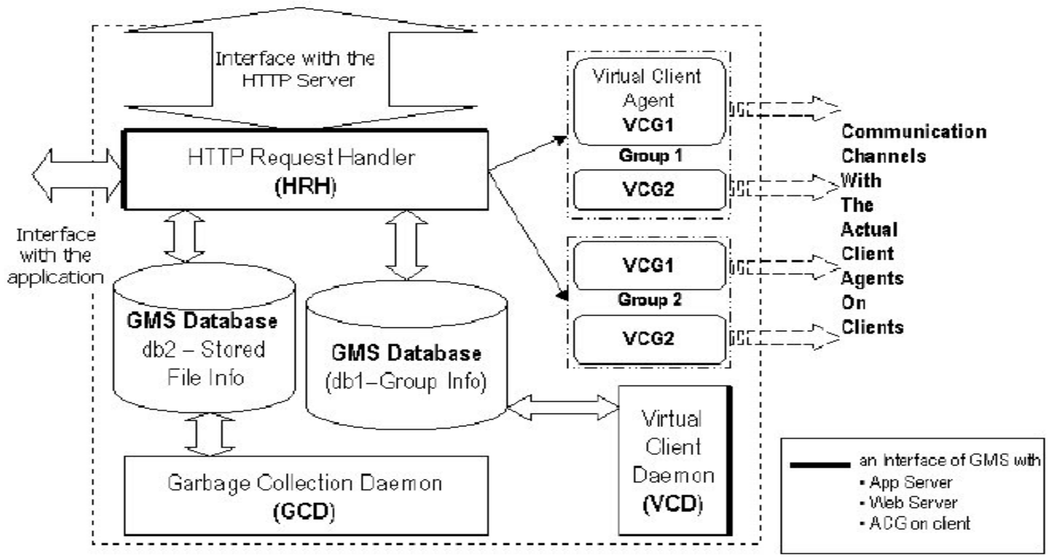
Interface with the HTTP Server
HTTP Request Handler (HRH)
Interface with the application
GMS Database db2 – Stored File Info
GMS Database (db1–Group Info)
Garbage Collection Daemon (GCD)
Virtual Client Daemon (VCD)
Virtual Client Agent VCG1
Group 1
VCG2
VCG1
Group 2
VCG2
Communication Channels With The Actual Client Agents On Clients
an Interface of GMS with
• App Server
• Web Server
• ACG on client

Fig. 3. Anatomy of GMS

D. *Identifying the requests made for group activity*

In HTTP-based communication, client requests are made by sending URLs to web servers. The proposed system too uses URLs to identify the request. All the requests corresponding to group communication will have a group pad. It contains information pertaining to the Group, the particular group member, various security related information, contains the path of the GMS (so that GMS gets the control before the actual application server) and other information pertaining to group activity.

The group pad is inserted into the documents before the response is sent to the group members. This is so that when these modified URLs (containing the group pads) are subsequently accessed by the users, the GMS knows the group to which the request corresponds to and can hence send the response to the other group members. Also, this pad is extracted and removed from the requested URL before forwarding it to the application server, as the application server is unaware of the groups.

URL Categorization: The URLs in the system are categorized using two different categorizations, as explained below:

1. *Master and Slave Requests:* The requests from the browsable area (section II) of clients are called *Master's Request*. Corresponding requests from all other members for the same document are called the *Slaves' Requests*. Master requests are forwarded to the application server and the slave requests are handled by the HRH itself. The corresponding clients are called master and slave respectively.

2. *HTML - URL Categorization:* For the purpose of the proposed system we have divided URLs appearing on HTML pages into two different categories. This is because both of them will have to be dealt with differently.

(a) *HREF Type URLs:* These URLs are those, which are included in HTML tags like ANCHOR, ACTION etc. in form of HREF or ACTION. To access the resources corresponding to these URLs, the user has to trigger certain events (e.g. clicking on the links corresponding to HREF or pressing the 'SUBMIT' button in a form). Only then does the browser send a request corresponding to these URLs.

(b) *SRC Type URLs:* In general almost all Internet documents have some objects like the images, image maps, style sheets etc., which are brought by the browser in subsequent connections with the server automatically. Such objects are specified in the Markup Languages by using the SRC, BACKGROUND, and HREF etc. in tags like IMG, BODY, and LINK etc. When the browser comes across these tags, the browser makes a request for the associated URL without the physical user asking to do so. We call these URLs as the 'SRC-type URLs'.
The basic difference in the two types of URL references is that the former represents a transition to a new document whereas the later represents some objects that are needed to complete the current document itself. The system therefore has to handle both of them differently. Different actions are also taken depending on whether the request is a master request or a slave request (refer section V-B)

Types of URL in the system: Given above categorizations, there exist at least 4 types of URLs viz. master requests contain both HREF-type and SRC-type URLs. So also have slave requests. Apart from these, there exists another special URL, used for the initial group formation (section V-A). Thus, there exist 5 types of URLs in all, as noted below (for detailed structure, refer Table I).

Type 0: These requests correspond to the initial stage of group formation, when the user wants to join or start the group. These requests are no way related to the application and are provided by GMS solely for the group management.

Type 1: URLs corresponding to master requests of HREF type URLs. The documents corresponding to the URL are stored in the local storage with the GMS, certain modifications are made to the databases, and the VCGs of the other group members are informed about the new request made by a master.

Type 2: URLs corresponding to slave requests of HREF type. These requests are for documents that had earlier been stored in the local storage.

Type 3: URLs corresponding to master requests for SRC-type files. These files are got from the site and sent as the response. A copy of them is also maintained in the local storage for subsequent slave requests.

Type 4: URLs corresponding to slave requests of HREF type. These requests are for documents that had earlier been stored in the local storage.

V. Working of the proposed system

The working of the proposed system is broken up into three different parts namely the *Initial Group Formation Model*, the subsequent *Transmission Model* and the *Security Model.* Initial Group Formation includes the process by which a user can start or join a group. Transmission Model includes the broadcast of the response of application server for each member to all other members of the group. And the security model includes the various security protocols incorporated to ensure that there would be no security breach in the access of a site due to the presence of the proposed system. The initial group formation and transmission models are discussed next. The security model is discussed in details in section VI.

A. Initial Group Formation/ Joining Model

The HRH handles all activities using URLs. For the initial group formation the system uses a URL type 0 (section IV-D). The system may use any authentication scheme as provided by the site, for example username/password, digital signatures etc. The current implementation uses the conventional username/password [18–21] interface for the authentication of the users (step 1, Figure 4). The system first authenticates the user, and if the username and password match, the system from here on believes that the user is the one whose account the server has.

Once the user is authenticated the system gives him an option, whether he wants to start a new group or join an existing group (step 2, Figure 4). If he wishes to join an existing group, the information about the groups is displayed to him and he chooses his group.

The existing group members of the group are then asked whether they want the particular new user to join their group. The group members are given certain identification information about this new user. The new user is admitted as a part of the group if the responses from other members satisfy the *group-joining criterion.* The group-joining criteria could vary, e.g. if 50% of the users give permission, then the user is allowed to join the group. Corresponding updates are made in the database (step 2.1.1, Figure 4). However, if the user started a new group, he would simply be assigned to the new group, and information about this new group is added to the database (step 2.1.1, Figure 4). After the addition of client in an existing or newly created group, the ACG is sent to the client. After reaching to the client side the ACG activates itself and tries to connect the VCD. VCD then creates a new VCG on the server side and binds it to the ACG (step 3, Figure 4). All the future communication between GMS and user/ client would now happen between VCG and ACG.

Once VCG has been created, GMS sends the URL of the *'Home Page'* or *the Initial Page'* and the *'Permission Granted Message'* to the user via VCG. ACG on receiving this message asks the browser to fetch the Home Page in the users *browsable area.* The newly added member is then given the status of the rest of the members so that it can start the group activity. The user gets the pages of other users in the corresponding read only windows (step 4, Figure 4). This is explained in Figure 4.

Leaving a Group: If a client wants to leave a group he informs it to GMS, who then updates the group information and the databases. After updating the databases GMS, removes the corresponding VCG and asks the other VCGs to get their clients informed and re-adjusted.

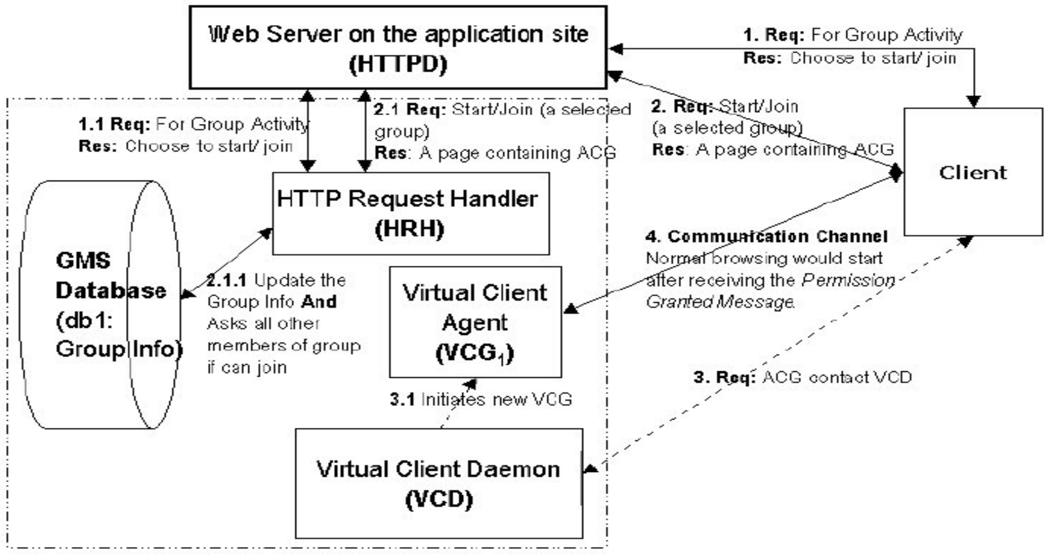
Web Server on the application site (HTTPD)
1. Req: For Group Activity
Res: Choose to start/ join
1.1 Req: For Group Activity
Res: Choose to start/ join
2.1 Req: Start/Join (a selected group)
Res: A page containing ACG
2. Req: Start/Join (a selected group)
Res: A page containing ACG
Client
HTTP Request Handler (HRH)
GMS Database (db1: Group Info)
2.1.1 Update the Group Info And Asks all other members of group if can join
Virtual Client Agent (VCG1)
4. Communication Channel
Normal browsing would start after receiving the Permission Granted Message.
3. Req: ACG contact VCD
3.1 Initiates new VCG
Virtual Client Daemon (VCD)

Fig. 4. Initial Group Formation/Joining Model

Rejoining a Group: In this case, the client agent who has the Group ID and corresponding secret key reestablishes the socket connection. This is also used for auto maintenance of the connection.

B. Transmission Model

This subsection describes the functioning of GMS after the group has been established and the normal browsing has started. The algorithm which GMS uses to process the various types of URLs and hence to manage groups is summarized next along with Figure 5.

When GMS receives a URL, U, it does the following:
• Verify the integrity of U (refer section VI-A).
• Determine the type of U.
– If U is of Type 1, i.e. a Master HREF, then GMS constructs the actual (non group enabled) URL by removing all the information pertaining to group. It then sends the request to the application server.
While *processing the response* received from the application server, GMS not only customizes the response for Master Client but also saves a separate and customized copy for the slave clients. For the master client it changes all the HREF type URLs to Type 1 URLs and all the SRC type URLs to Type 3 URLs with same group and client ID. Before changing the SRC type URLs to Type 3 URLs it creates an entry in the File Database, db2, and uses the index of the entry in the Type 3 URL construction.
In the customized copy saved for the slave clients it disables all the HREF type URLs and converts all SRC type URLs to Type 4 URLs. The index of the file entry used in the creation of the Type 4 URL is same as the one used in the construction of Type 3 URL for Master response. The group ID is the same. Since the client ID is not known at the time, client information is not filled and is added on the fly while serving the document.
After processing and saving the document, GMS sends a broadcast message to all the other members of the group for requesting the document. It constructs Type 2 URLs for it and sends to ACG through VCGs.
– If U is Type 2 the request is not forwarded to the application server, instead the document from the local storage is retrieved. All the Type 3 URLs those were incomplete while saving are completed with the client information and served. The send bits are set in the database for this file.
– If U is Type 3, then the actual URL is constructed by removing all the group related information and forwarded to the application server. The response is sent to the master client and a copy is saved in the local storage (in the path given the db2 entry for the index). The send bits are updated to indicate the document has actually been received from the application server. It then notifies all the ACGs that have already requested for the document.
– If U is type 4, then the document present in the local storage is served. If no document is present then it waits till a time-out happens. In case of time out a notifier is added to notify the ACG of this client whenever the document is brought in local storage.

VI. Securing - the proposed Group Browsing

The system discussed so far is generic in nature. It can be plugged on to an existing application to make that application accessible in a group. But as the type of applications varies over a wide range, with each application having its own requirement, the security requirements of each of the application is different. The system will not plug any security hole in the implementation of the application but has to make sure that no security loophole occurs due to its presence.

Below we describe certain security hazards which might arise due to group enabling a site and later we propose the security model which takes care of all the below mentioned loopholes.
• The users should be authenticated (refer section V-A) i.e. proof of identity must be established.

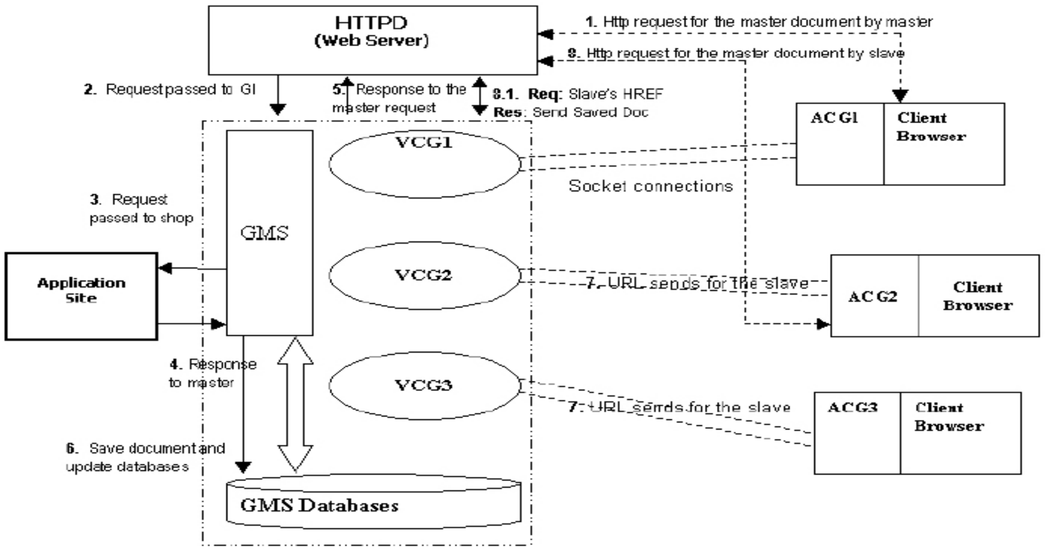
HTTPD
(Web Server)
1. Http request for the master document by master
8. Http request for the master document by slave
2. Request passed to GI
5. Response to the master request
8.1. Req: Slave's HREF
Res: Send Saved Doc
VCG1
VCG2
VCG3
ACG1
Client Browser
ACG2
Client Browser
ACG3
Client Browser
Socket connections
3. Request passed to shop
GMS
Application Site
7. URL sends for the slave
7. URL sends for the slave
4. Response to master
6. Save document and update databases
GMS Databases

Fig. 5. Transmission Model

- GMS sends the actual client agent (ACG) to the client when the client has successfully joined or started a group. If an intruder can replace the ACG with a customized ACG while it is being sent then the system is breached. So, the ACG should thus be sent over a secure channel.
- An intruder should not be able to record and replay any transaction. Such replay-attacks would make the site vulnerable to a lot of damage by intruders.
- The messages being exchanged between the VCG and ACG should be authenticated and their integrity should be maintained. Without this the intruder can misbehave in various ways e.g. send fabricated-messages like logout etc.
- During the activation of ACG, when ACG contacts the VCD, the intruder should not be able to hijack the connection and hence being able to proxy as the Server.
- The response sent by HRH should be authenticated. If it is not then all the group members may not see the same things. The intruder would be able to show different responses to different members.
- Even the group members should use only the links provided by the system and should not be able to reconstruct or tamper with the URLs. No one should be able to construct an URL that corresponds to a group activity. Integrity of the URLs should be maintained.

A. Proposed Security Model

The proposed security model provides authentication and prevents the malicious replay attack. All ACGs have the following random numbers: n_1 to expect from GMS on socket, m_1 to send to GMS over socket, N_{1i} to expect from GMS on HTTP for each the ith client of group and M_{1i} to send to GMS on HTTP for each of the ith client of the group. The ith client for an ACG corresponds to the read only window with the ACG for the ith client in the group.

Since in the socket communication, all the messages are transported between the ACG and the VCG over the same socket, integrity of the messages can be maintained by a pair of random numbers (section VI-A.3). But in case of HTTP communication, since each browser window of client (there are multiple browser windows on the client corresponding to the members of the group, see section III) generates a separate HTTP request, the response and requests of different windows are independent, thereby mandating use of different random numbers. In case if we use the same numbers then there would be problems for instance while one request has been sent other request could not be sent concurrently.

Each client shares a secret key with GMS, K^{shared} (it is termed as share, because it is shared the clients and the GMS). This key is stored in the Client Record of Group Database, db1. It is exchanged during the initial authentication (using SSL). GMS has another secret key, K^{secret}, but only GMS know this.

A.1 Preventing URLs from being tampered

In order to prevent the tampering of URLs, the URLs would contain the keyed hash [26] of the rest of the URL appended at the end of the URL. Thus when a request corresponding to a URL comes to the server, the GMS first checks for the authenticity of the URL and only then does it further process the request. The key for the hash is private to GMS (K^{secret}). This ensures that even the group members cannot reconstruct or tamper with the URL.

A.2 Authentication during initial group formation

It uses the conventional username/password [18–21] interface for the authentication of the users. Once authenticated the system then believes that the user is the one whose account the server has. If the application supports an authentication scheme, then the authentication by the GMS for the group may be merged with that of the application. In order to provide security during the initial

Authentication Pad: <Message>$/n_1 + 1/n_2/$ Keyed Hash
n_1 = Expected number, n_2 = Next number to expect
Keyed Hash = Hash of (<message>$/n_1 + 1/n_2$) with K^{shared}

Fig. 6. Authentication Pad

group formation and the actual client agent transfer, the system uses Secure Socket Layer Protocol (SSL) on HTTP (i.e. HTTPS) [12–15]. The user name and password from the client are sent to GMS on HTTPS.

When the ACG is sent to the client over SSL, a Shared Secret Key (K^{shared}), the client's Group ID, Client ID and the other nonces as mentioned above i.e. n_1, m_1, N_1, M_1 are also passed to the ACG. When ACG initializes and connects to GMS (or more precisely the VCD), it authenticates itself by sending the *'ACG Authorization Message'* to the GMS. The message consists of the GID, CID, m_1+1, a new Random Number (m_2) and the Keyed Hash of the tuple, the key being the shared key (K^{shared}). Since only ACG has K^{shared} and m_1, the above-mentioned tuple unmistakably identifies the ACG.

A.3 Authentication of Requests and Responses

When a client, whether a master or a slave requests GMS for a document, GMS has to verify that the request is coming from a valid client. This is because a malicious intruder might request for a dummy page as a master, and then GMS would distribute the same page to the other members of the group. This is clearly undesirable and hence the need for authentication of the requests. Also an intruder must not be able to record a transaction and replay it backs again at a later time thereby be fooling the group members.

In order to prevent the replay attack, system takes the approach of maintaining pair of random numbers for each expected request/response and refreshing the numbers in subsequent calls. As assumed in the design of the security model (section VI-A), all ACGs have a pair of random number for communication on the socket and a pair of random numbers for each client for HTTP communication. The initial nonces are sent to the client, along with the ACG. Since the initial communication is on SSL, the initial numbers are exchanged safely.

The approach taken in the system is to append an authentication pad after the message in case of the socket or after the URL in case of HTTP. The authentication pad is shown in Figure 6. Since the party on the other side, be it GMS or ACG, knows the expected number the replay attack can be prevented.

• **Authentication of messages over socket:** When the VCG sends a URL or for that matter any message to the ACG over the socket, it appends the authentication pad with the message. The numbers used in this case are n_1 and m_1, respectively for GMS to ACG and ACG to GMS communication.

• **Authentication of HTTP Requests received by GMS:** GMS processes the requests it receives in much the same way as explained in Section V-B. Besides the normal processing explained earlier, it takes some extra steps to verify the authenticity of the URL and makes some modifications in the generated URLs to prove the authenticity to the ACG.
As mentioned in Section VI-A GMS maintains a set of random numbers for each ACG, where each set has a random number corresponding to a member in the group. The expected number is changed to the next expected number only if the request is a request for some other document. Since a document can be split across over multiple HTTP requests as in case of SRCs (see section IV), the expected number for such parts of documents (images, maps etc.) is same as the expected number for the main document.
Thus the expected number is only changed if there is a Type 1 of Type 2 request. The SRC requests (Type 3 and Type 4) do not change the expected number. The formats of URLs expected by GMS are shown in Table I.

• **Authentication of Response over HTTP:** In case of the HTTP response, GMS passes the authentication pad as part of the HTTP header. The authentication pad contains the keyed hash of normal response appended with $N_1 + 1/N_2$. The ACG verifies the authentication by recomputing the Keyed hash. It then changes its next expected number to N_2.

VII. Implementation issues and analysis

Performance analysis of the system is done to check the feasibility of the system on the net. A Pentium II based PC with 128 MB RAM (100 MHz bus speed) and loaded with Windows NT Workstation with Service Pack 6, Personal Web Server (Microsoft's Web Server) is used for prototype implementation and performance analysis. The tool used is the application "Apache Benchmark" for making given number of requests to the server. Apache benchmark was run from two different machines running Red Hat Linux v6.2. The performance is found to be satisfactory. It is observed that response time of the server with GMS (i.e. browsing in a group) is four to six times more than response time of the server without GMS (i.e. browsing individually). Response time of the server with GMS depends on the following; (*i*) load on the server, (*ii*) number active groups and (*iii*) number of members in each active group.

The performance can be improved upon by taking care of the following implementation issues:

• Implementing the GMS as a servlet (in case of Java), which is pre-loaded in memory, instead of a process being forked each time a request arrives at the web server (e.g. CGI scripts). The GMS could also be implemented as a *filter* in the W3C Jigsaw Java Web server [22].

• If the GMS sent the entire document, to the ACG, over the socket instead of just the URL of the saved document in the local storage, client side latency would be reduced to half the round-trip time to the server. The problem with this design is that there is no way the ACG (an applet) can display a page (i.e. parse it and display it), which has been got on a socket, on the common browser windows. In the Netscape Navigator (v6), the applet has more control over the browser and the said design may be a possibility. But

TABLE I: URL Types

Type	Description	Format
Type 0	Group Management	https://<Server>/<GMS>/<ManagementInfo>/0/Hash$_{k^{shared}}$
Type 1	Master HREF	<Protocol>://<server>/<GMS>/<documentPath>/<GID>/<CID>/1/ M_1+1/M_2/Hash$_{K^{secret}}$, where M_1, M_2 are for this ACG and this window.
Type 2	Slave HREF	<Protocol>://<server>/<GMS>/<indexOfStoredFile>/<GID>/<CID>/2/ M_1+1/M_2/Hash$_{K^{secret}}$, where M_1 corresponds to this ACG and M_2 is for the window of ACG of the Master of this Document.
Type 3	Master SRC	<Protocol>://<server>/<GMS>/<documentPath>/<GID>/<CID>/3/ M_1+1/M_2/Hash$_{K^{secret}}$, where M_1, M_2 are for this ACG and this window.
Type 4	Slave SRC	<Protocol>://<server>/<GMS>/<documentPath>/<GID>/<CID>/4/ M_1+1/M_2/Hash$_{K^{secret}}$, where M_1 corresponds to this ACG and M_2 is for the window of ACG of the Master of this Document.

this possibility hasn't been explored.

- File handling is a very expensive operation. Keeping the recent files (up to a certain maximum size) in memory, could improve performance.
- The database implemented was a simplistic one. It used mutex locks for per-record locking. It didn't maintain any index of the table according to the primary key of the table, thus increasing the search time. The search being linear took time in the order of $O(n)$. This search time could be improved to $O(log(n))$ by using appropriate search algorithms and by maintaining indexes on the primary fields. Other database optimizations will improve the performance of the system [23].

VIII. Conclusion

The design of a generic system to support group browsing as proposed in this paper is a pioneering effort. It proves that *'The Concept of togetherness'* and *'Group Activities'* are now no more unachievable.

With this noble effort a new horizon can be opened for the Internet Community. We may soon have, Group Shopping Malls, Group On-line Medicare, Group On-line Education System and many more.

References

[1] Peter Stone, Michael L. Littman, Satinder Singh, Michael Kearns, "ATTac-2000: An Adaptive Autonomous Bidding Agent", *Journal Of Artificial Intelligence Research 15*, pp. 189-216, 2001.

[2] H. Schulzrinne, "RTP Profile for Audio and Video Conferences with Minimal Control.", *RFC 1890*, Audio-Video Transport Working Group, January 1996.

[3] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1", *RFC 2616*, June 1999.

[4] M. Kumar, A. Jhingran, R. Anand and R. Mohan, "Future of Sales Promotion on the Internet", *Proc. 3rd USENIX workshop on Electronic Commerce*, Boston, September 1998.

[5] Laurie Law, Susan Sabett, Jerry Solinas, "HOW TO MAKE A MINT: THE CRYPTOGRAPHY OF ANONYMOUS ELECTRONIC CASH", *National Security Agency, Office of Information Security Research and Technology, Cryptology Division*, 18 June 1996.

[6] N. Asokan, Phil A Janson, Michael Steiner, Michael Waidner, "State of the Art in Electronic Payment Systems" *IEEE Computer*, volume 30, no 9, pp. 28-35, 1997.

[7] K. Bauknecht, S.K. Madria, and G. Pernul (Eds.), "EC-Web 2001", *LNCS 2115*, pp. 81-90, 2001

[8] ePSO (Electronic Systems Observatory), available at http://epso.jrc.es/

[9] DigiCash. See http://www.digicash.com/ecash/

[10] See www.hdfcBank.com, www.CitiBank.com

[11] See www.amazon.com, www.ebay.com

[12] A. O. Freier, P. Karlton, and P. C. Kocher, "The SSL Protocol, Version 3.0", *Internet Draft*, March 1996.

[13] K. Hickman and T. Elgamal, "The SSL Protocol", *Internet Draft*, 1995.

[14] Netscape Corporation, "Understanding Encryption and SSL", Chapter 14, http://developer.netscape.com /docs/manuals/proxy/adminux/encrypt.htm

[15] Joris Claessens, ark Vandenwauver, Bart Preneel, Joos Vandewalle, "Setting up a secure web server and clients on the internet", em IEEE (0-8186-8751-7/98), 1998.

[16] Ran Canetti and Hugo Krawczyk, "Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels", in *proceedings of Eurocrypt 2001*, LNCS 2045, 2001.

[17] C. K. Wong, M. Gouda, S. Lam, "Secure Group Communication using Key Graphs", *IEEE/ACM transactions on networking*, Vol. 8, No. 1, 2000.

[18] S. P. Miller, B. C. Neumann, J. I. Schiller, "Kerberos: An authentication service for open network systems", in *Proceedings USENIX Winter Conf.*, Feb 1988.

[19] R. Rivest, "The MD5 Message Digest Algorithm", *RFC 1321*, April 1992.

[20] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", *RFC 2617*, June 1999.

[21] W. Stallings, "Cryptography and Network Security, Principles and Practices, 3/e", *Prentice Hall*, 2002.

[22] Jigsaw, *W3C Java Web Server*, details available at www.w3.org/Jigsaw/

[23] G. Couloris, J. Dollimore, T. Kindberg, "Distributed Systems Concepts and Design, 3/e", *Addison-Wesley*, 2001.

Appendix A: Scemas of Databases Used

Database	Record	Constituents
db1 Group Database	Group Record	Group ID and Client IDs
	Client Record	Client ID, VCG Pointer, User Information and share Secret Key (k_{shared})
db2 File Database	Group Record	Group ID and File Indices
	File Record	File Index, Master Client ID, Local File Path, List of Members present at the creation time, Send bits for the members