SwanLink: Mobile P2P Environment for Graphical Content Management System

Popovic, Jovan; Bosnjakovic, Andrija; Minic, Predrag; Korolija, Nenad; and Milutinovic, Veljko

Abstract—*This document describes major trends in the present, and what authors reckon as the future of distributed computing and online collaboration. It considers topology and communication protocols which will be used as standard in future distributed application. Analysis of everything that has spawned from Fred B. Holt and Virgil Bourassa original idea concerning methods for online collaboration based on regular graph topology is presented. After introductory part on current implementation and techniques related in this field, our solution is presented through the project which is a practical implementation of the ideas presented in this paper. Facts and dogma about modern computer infrastructure, communication protocols, and its reliability and scalability are discussed here. Authors' point of view and vision of future work in this field of distributed application is also included.*

Index Terms—*calm technology, context-awareness, pervasiveness, ubiquitous computing*

1. Introduction

THIS document is about humans' all increasable need to communicate with other regardless to space distance that divides them. Increasable grow of telecommunication technology and internet give the people opportunity to communicate with each other in most various ways. Classical communication methods such as e-mail messages, seems to be just satisfying in first era of communication technology. In the first generation of communication technology sending messages to friends and waiting for them to answer was enough for most of the users. In those kinds of systems simple client-server architectures, where one entity has just send request to server (i.e. e-mail message), waiting them to fulfill his need was developed, was quite stable and has done its purpose very well. But increasable desires, such as real time communication with complex multi-medial content, bring many problems to the distributed system architecture. Problems which need to be solved in new architecture were:

- Bandwidth,
- Scalability,
- Stability,
- Mobility.

Those kinds of problems can not be solved using traditional client server approach. Client server philosophy is based on assumption that we have one powerful computer (i.e. mainframe) which serves tens or hundreds small clients. Increasable number of personal computers, with even increasable power, low cost, and availability to ordinary man, opens the new world of opportunity for techniques and protocols which can be used to interconnect them. One of the most promising infrastructures which indicate that will be standard used in most of the future application is peer to peer (P2P). P2P is philosophy where lots of independent computers collaborate with each other, regardless their power and location. In following sections we will describe how P2P architecture can be used to solve a problem of cache consistency in distributed applications.

2. Problem Statement

Theme of our research was development of distributed application which enables complex graphical context exchange between users regardless their distance and number. Application should be able to support large number of participants that can concurrently modify common unique content.

Graphical applications are the most complex type of content management systems regarding the various type and size of the elements, which need to be exchanged with other participant. Due all participants have content cached in local, one of the biggest problem was how to maintain cache consistency between them asynchronies without forcing them to ask for refreshing their local content.

Philosophy we use here is a little bit different from classical push-pull methodology which is used in traditional internet approach, which is currently used on the internet, where the user explicitly demands for update their local content.

Our problem was how to develop fast, scalable and reliable infrastructure and protocols which can be used for maintaining local views for all participants, without their interference in synchronization process.

Manuscript received April, 2006. This work was supported in part by the Panthesis, Inc., USA.

Popovic Jovan, Bosnjakovic Andrija, Minic Predrag, Korolija Nenad, and Milutinovic Veljko are with the Faculty of Electrical Engineering, University of Belgrade, Serbia.

In order to do that, the next problems had to be solved:

- managing user joining and leaving,
- keeping consistent cache state,
- deleting objects,
- sending messages.

3. *Existing Solutions and Their Criticism*

In the history we can find many attempts to implement online collaboration between distance participants. Distributed architecture as we know it today, had several major trends in the past several decades regarding relationship between computer(s) and person(s) using it (them).

Client-server solution appeared as a very first solution, obviously ideal for the set of problems that existed in the very beginning of the network. In the very beginning of computing, only mainframe computers existed, and we have bunch of small terminals using their services to fulfill their needs. This approach was predecessor of Client-server architecture that currently being used.

Collaboration between clients has been established via central server using hub and spoke methodology - client sends message to server and server broadcasts the message to all other clients. In this kind of architecture clients were responsible for handling the messages received.

Those kind solutions were appropriate for small and compact communities that are localized on geographically small area, and it is inappropriate for highly reconfigurable and mobile internet connected communities, which currently exist. Possibilities for real time interaction between clients in client server middleware can be accomplished using so called Hub and spoke message-passing techniques, which is showed on Figure 1. In Hub and spoke system, one client send the message to the server which is responsible for establishing the communication between clients, that server broadcasts the message received to all other participants.

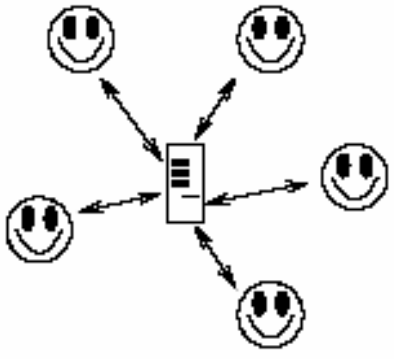

Figure 1: Hub and spoke techniques used in client server middleware for real time participant interaction

The major problem with client server (or hub and spoke) architecture in area of real time online collaboration lies in the fact that this solution is absolutely not scalable. Namely, in communities with thousands participants who collaborate mutually via one server, we can see that server represent the bottleneck of the system. Regarding the issue that in client server system we have N clients on one server, complexity of such kind of system grows as N^2. Due to large number of clients, often server cannot dispatch all received messages from one client to all others, which is unacceptable in modern real time collaboration environments.

Additional problem lies in the fact that client server architecture is mentioned for relatively static communities. Although clients can be placed on various locations, server is (in most cases) placed on static location which is good known to all users. That kind solution is unacceptable in mobile communities where clients can migrate across the world without affecting the session.

Due to mentioned characteristics, client server architecture wasn't so popular in real time application where large number of participants needs to be supported. Client server architecture is effectively used in simple request-response systems, where number of clients can update his local copy of document, post it to the server when is done and retrieve the newest version on demand.

One of the solutions for improving scalability in client server middleware, which is often used, is server farm implementation. Server farm is set of (more or less) equal server instances that cooperate mutually so the client's request can be processed more effectively. Basic idea is that server can mutually redistribute responsibilities for processing client's requests. In contrast to pure client-server solution, where one server cooperates with N clients, in server farm environment we have K server, which mutually collaborate and serve participants request. In server farm system number of participant is partitioned in K groups where one group is under responsibility of one server in farm. Scalability is little bit improved but still it is not satisfying for our solution.

Although server farm has many disadvantage in scalability and supporting the large number of clients, it has found purpose in the effectively online gaming support local network with medium number of participants. Lot of online gaming system has to be implemented using this technology because it supports clients which do not permanently come and go, so scalability issue does not come in the first plan. In our solution, where large number of participants constantly comes and goes, server farm technique can not be applied.

Only solution, which currently exists and can satisfy requests for scalable, real time collaboration infrastructure, which can be easy

reconfigured, is Peer-to-Peer solution (P2P). P2P environment contain large number of collaborator with equal rights. That kind of solution is completely applicable for most of our needs, because it promises that it will support large communities.

4. PROPOSED SOLUTION

4.1. Proposed Solution in General

SwanLink represents revolution solution in area of distributed mobile P2P application that provides users with capability to exchange complex graphical elements with each other. It is a multi-user environment which enables users to concurrently create and modify common graphical content. Graphical content represents set of graphical elements such as lines, pictures and text which can be managed through the system interface. Graphical content is shared among all participants who are currently involved in document managing session, using the local copy stored at each participant's computer. Contents of those copies need to be consistent although they are cashed in physically different locations, without forcing the user to explicitly require synchronization with other participants. Due to such kind requirements system itself needs to take care about maintaining participants' local cache consistency instantly whenever anyone makes some change. Whenever some of the participants alters content on his local view, message that carries information about change that is made, is broadcasted to all others participants. All other participants update theirs local content instantly when they receive notification about modification.

Graphical content is not explicitly owned by any of the participants who are involved in session. Everybody has the equal rights regarding the content management. Newcomers who join the session afterwards will fetch valid copy of current graphical content and be able to work on it just like old ones.

This feature is the key for establishing mobile and secure community for graphical content management. Due to issue that content is saved on EACH participant's computer, chance that it will be lost or destroyed is possible just in case that all participants simultaneously fail down. As long at least one participant "lives" he will preserve valid copy of session's content which can be shared with other future participants.

Regarding the issues mentioned above we can see three key requirements which need to be provided with SwanLink:

1. Ability to exchange and share graphical context regardless the participants which are currently in session.
2. Ability to background maintain graphical content.
3. Ability to preserve common content until at least one participant is present.

Regarding the issues mentioned above the most important part of system is network tier which can support those features. Network tier needs to provide easy reconfigurable community with ability to safely handle leaving from community and acceptance of new participants without possibility to whole community departure. Also network tier needs to provide fast message passing. In following section issues regarding the sufficient architecture of network tier are discussed.

4.2. Proposed Solution for Concrete Situation

Even we have selected P2P architecture as the best one for our solution; our job was still not done. Between various P2P implementation which are currently implemented we needed to find one that will completely satisfy our requests.

One of the frequently implemented solutions for P2P communities is all-to-all architecture. In all-to-all architecture each participant is directly connected with each other as shown in Figure 2.

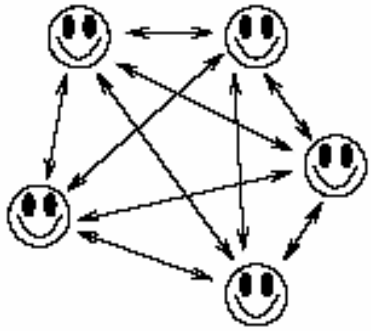

Figure 2: Peer to peer network implemented as all to all community

Advantage of this solution lies in the fact that all messages are immediately delivered to all other participants, so the latency is minimized. Disadvantage of all-to-all solution is that it is very hard to maintain those kinds of communities. Reason for this is the fact that newcomer, which arrives in community, needs to find ALL existing participants so he can join the group. Another problem is frequently connection changing that needs to be established with other participants. This solution is satisfying for small communities where we have not frequently architecture change.

Another extreme approach is daisy-chain implementation showed in Figure 3. In daisy-chain implementation number of connection which is established between participants is minimized, prejudice message delivery time.

In daisy chain, message is sent from one participant to its nearest neighbor, and the message is forwarded to next level of peers. Obviously, problem in this kind of solution is unacceptable latency of message. Another reason why this architecture is not applied, is reliability of that kind of P2P community. Namely, if one of the participants fail down entire community is partitioned in two groups, without possibility to heal them self.

Solution which is probably the most acceptable for underlying communication infrastructure may make use of a multicast network protocol, or a graph of point to- point network protocols, or a combination of the two, making some kind of balanced spanning tree.

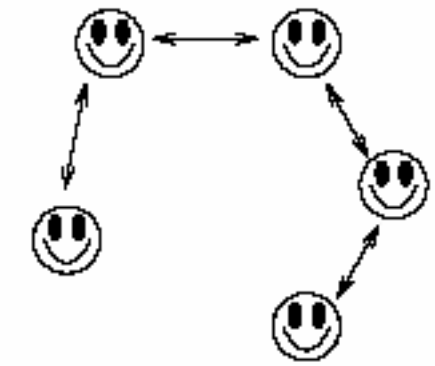

Figure 3: Peer to peer network implemented daisy chain community.

That kind solution provides logarithmic message delivery latency as number of participant grows. The infrastructure in commercial use today, in products such as IBM's Sametime, Data Connection's DC-Share, and Microsoft's NetMeeting, is the T.120 Internet standard.[swan_white_paper].

T.120 Internet Standard: When first connecting to a T.120 communication session on a given host computer, the local application spawns both a proxy process (the MCU) and a daemon process. The daemon process is a resident process which instantiates the MCU and listens for additional requests. The MCU forms a direct connection to the MCU of another host designated by the application user, or is designated as the root of the session. The requesting process and all additional processes on the host wishing to join the session, form a direct connection to the MCU process on that host. To share information, a process sends a message to its MCU, which is sent up the tree of MCUs to the root, then down the tree of MCUs and disseminated among their attached processes.

Although T.120 gives many benefits in distributed on-line collaboration, many disadvantages can be seen regarding its healing and reliability issues. The major problem in spanning tree implementation lies in the fact that administration network topology lies in end user responsibility. When one of the nodes in tree departure, it is responsible for interconnecting all his children to highest level so they will not be departure from parent nodes. In case of disgraceful node fall down, all child nodes will be lost.

Solution that provide logarithmic message latency and reliable infrastructure which is required in our project needs to be based on regular graph topology. Among various graph topology we have choose regular graph topology. Regular R-connected graph represents topology where each vertex is connected with fixed number of neighbors. For regular graph we say that is r-connecter, where r is the number of his neighbors. Example of 4-connected regular graph is showed on figure 4.

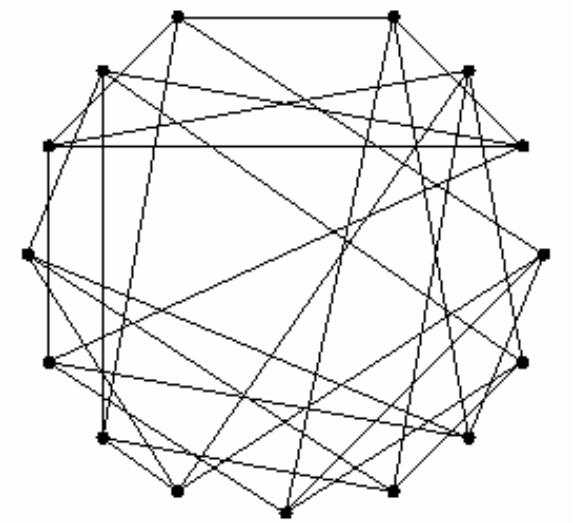

Figure 4: Regular graph. In current example each vertex is connected with exactly 4 nodes.

One of the biggest problems in distributed communities is preventing the cascade departures of nodes when the node that was represent the connection between two group of nodes in community (i.e. in tree topology when parent node fail all the children will be departures from community because the failing one was the only connection).

The key benefit of regular graph community, that exceeds this problem, lies in the fact that every node at each time is connected with exactly r-nodes which give whole community reliability against departures issue. In case that one of the node fails his neighbors will be still strongly connected with community via other three neighbors. Also in case of node failure four nodes which lost one neighbor (the failing one) have ability to interconnect mutually and again establish regular graph infrastructure. Hence, we can find remarkable self-healing opportunity in communities organized as regular graphs. In worst case scenario if four nodes fail which surrounds one neighbor simultaneously fail down as side effect we will have only one node departure from community. Regarding the low probability that the right combination of failing four nodes occurs, we can say that kind of collateral damages will be very rare.

5. *ANALYTICAL ANALYSIS*

Swan (Small-world Wide Area Networking) is a sophisticated implementation of a revolutionary theory of distributed logical networking developed by Virgil Bourassa and Fred B. Holt. Swan was developed to transparently and interactively

communicate, via ad hoc sessions, across intranets and the Internet between hundreds or even thousands of computers (as true peers) while maintaining minimal latency and high security. Features of Swan include very high reliability via adaptive self-healing, minimal latency via active self-optimization, high security via strong authentication and encryption, and virtually no administration or maintenance overhead.

We have used Swan technology because it is [4]:

1. Economical because distributed intelligence eliminates the need for expensive centralized servers to manage data or control flow.
2. Simple to use because it is transparent to every participant with virtually no administration required.
3. Easy to implement. Network communication is transparent to the application; the API is quick and simple to use.
4. Real-time responsive and eliminates the bottleneck of centralized servers and self-optimizes network data flow to achieve the most efficient organization available.
5. Reliable because it is self-healing and requires no hardware or infrastructure planning.
6. Secure because incorporates strong authentication and encryption.

This section is supposed to introduce the reader with the advantages of the above described network architecture.

First, here is the legend of this analysis:

n - number of nodes participating in the architecture,

p - probability that the connection between two nodes will fail (only in case of equal probabilities),

tindex - time needed for node index to respond,

tnodei – time needed for node i to respond.

The analysis is done in the following way. First we define three suitable network architectures which are basically different. Than we compare the main three network architectures with Swan considering time needed for communication, and than considering reliability, and then we try to use one more real model, which is based on the fact that the nodes are not equal considering speed and reliability. Finally, we compare the result to see what the basic differences between given architectures are.

1. Client-server architecture - in this architecture we consider that all of the user nodes are equal, and that they communicate with each other indirectly, by sending messages to the server.

2. Ring architecture - in this architecture we consider equal nodes that are forming a ring, which means that each node is connected only with his two neighbors. Communication is done by sending a message from the starting node to the next one, and so on, until it reaches everyone.

3. P2P all-to-all architecture - in this architecture we consider equal nodes that are forming network in which every node is connected to each other node.

4. Swan - in this architecture we consider equal nodes that form network where every node is connected to four other nodes randomly c

Time needed for finishing communication process:

Client server: n*(tnode1+tserver)

Ring: n * (tnode1+tnode2)

P2P: n * (tnode1+tnode2)

Swan - 4 * log n * (tnode1+tnode2)

What we can see is that advantage of Swan grows as the network grows. If the network consists of nodes that are not equal, it is obvious that all three architectures would have problem with waiting for it, and so much worse result considering time, while Swan wouldn't be affected, because the message will be passed by "the fastest node".

Reliability of the architecture:

Client server: if one node fails, only he will be disconnected, but if the server fails, all of the nodes would became disconnected. As the server is the most used, it would be normal to expect it first to fails. That leads to the fact that this architecture is not usable when we need system that "must not fail".

Ring: p*p*(n-1)*(n-3)+p*p*(1/n)*(1/n) - probability that one node will became disconnected from some other node is equal to probability that any two nodes fail, except if they are two neighbors plus the probability that it will fail itself.

P2P: If one node fails, only it is disconnected.

Swan: In theory, if four nodes connected to a node fail, that node would became disconnected (probability p^4). In practice, we have different situation: As mentioned above, when one connection breaks, nodes that were forming this connection immediately are starting to search for other nodes that have less than four connections. This way, even if more than half connection break, it would be possible to have all of the rest nodes connected to each other, having still four connections per node. As we can see, the Swan architecture is both the fastest in the set given above and the most reliable.

6. CONCLUSION

Having in mind everything mentioned in this paper, we can say with assurance that computer architecture defined here deserves to be one of the leading computer network architectures in the future. The main goal of obtaining both reliable and fast message passing led to success.

REFERENCES

[1] Milutinovic, V., "Some Solutions for Critical Problems in the Theory and Practice of Distributed Shared Memory,"

[2] http://www.computer.org/tab/tcca/news/sept96/sept96.htm

[3] Milutinovic, V., Stenstrom, P, "Improvements of Distributed Shared Memory: The Wholastic Approach," *Proceedings of the IEEE*.

[4] http://www.panthesis.com/features.php