Research and Development in E-Business on the Internet

Milutinovic, V.,

and

Cvetkovic, D., Aleksic, M., Davidovic, G., Dimitrijevic, Z., Dingarac, D., Grubic, M., Grujic A., Horvat, D., Horvat, Z., Ikodinovic, I., Ilic, J., Ilic, N. Jovanov, E., Jovanovic, M., Jovanovic, T., Knezevic, P., Krsmanovic, D., Marinkovic, G., Mihanovic, M., Milenkovic, A., Milicev, D., Milicevic, M., Miljkovic, V., Milutinovic, D., Mirkovic, J., Mitrovic, M., Nikolic, M., Petkovic, D., Petkovic, Z., Protic, J., Protic, S., Raskovic, D., Savic, S., Slijepcevic, S., Sokic, I., Stefanovic, M., Tartalja, I., Tomasevic, M., Tomca, N., Radunovic, B., Vuletic, M., Zgonjanin, S.

Department of Computer Science and Engineering
School of Electrical Engineering, University of Belgrade
http://rti7020.etf.bg.ac.yu/rti/ebi

Abstract

This paper describes R+D projects of professor Milutinovic and his associates (currently on-going or recently completed - on or before 1.1.2000.) in the field of system support for e-business on the Internet (EBI); details are available on request or through the papers published so far (see the author's WWW page and WWW page of the EBI research group). The stress is on issues of importance for HICSS-34.

1. Introduction

Internet is much bigger, older, and more efficient than we tend to think. It is about thirty years old, it supports more than 10TB (TeraBytes) of data flow daily, and it doubles (in almost every respect) each year. Impacts of Internet technology are multidimensional (social dimension and business dimension are stressed the most).

The term EBI (Electronic Business on the Internet) refers to a synergistic interaction of a number of disciplines, like electronic multimedia, electronic collaboration, electronic commerce, and appropriate business knowledge and intelligence infrastructure. There are several major infrastructure bottlenecks for efficient use of EBI. Some of the bottlenecks are in the hardware and others are in the software domain. However, that hardware and software problems are interrelated. Also, no bottleneck is only of the hardware type or only of the software type. Always one of the two components prevails, and that is why it is possible to classify the hot research topics. In the hardware domain, the major problems are how to bring bandwidth to users, and remote control of resources (which increases the capabilities and their performance). In the software domain, the major problems are how to do efficient information search for remote information (agents technology, genetic algorithms, simulated annealing, XML), and reuse (proxy caching, media caching), which decreases the bandwidth requirements.

These bottlenecks are, at the same time, the hot research topics of EBI (important bottlenecks typically turn quickly into important research problems), as described next:

2. Obelix: An accelerator for business search

Currently, if one does a search for the best item to purchase, the items found are ranked by the amount of keywords matched. However, this is not the only issue, and definitely not the most important issue, relevant for a successful business practice. What is more important is how are the other customers satisfied with purchasing from the same WWW site. Consequently, it is useful, before purchasing something from a WWW site, to check the general customer satisfaction with that site. For that to be done, one can collect information about: (a) the number of times the page was visited, (b) cached, (c) printed, (d) purchased from, etc. The visiting may be weighted with one point, caching with 2 points, printing with 3 points, purchasing with 4 points, etc. Once the first purchase by a given customer is done, a timer may be set. If the second purchase comes before some time interval, it can be weighted with 8 points (as an indication that the customer was happy with the first purchase). If no purchase within the prespecified time interval, the weight of the first purchase may be converted to -4 (as an indication that the customer was not happy with the first purchase).

Submitted to HICSS-2001, Maui, Hawai'i, USA, January 2001.

The plethora of possibilities is extremely wide and any scenario of the kind can be implemented. During some time (phase #1), the system would collect information about the number of outcomes of certain type. After that (phase #2), the system would generate the weighted sum for each URL that is being monitored. Finally (phase #3), the search database would be updated, so the next ranking can be devised as a combination of keywords match and customer satisfaction. In addition, the customer profile data can be brought into action, so different users obtain different ranking, for the same conditions, and for the same inquiry. For all above to be implemented, in addition to the PC which serves as an Internet provider, one may need hundreds of other PCs to implement the above described three-phase activities. An alternative is to implement the algorithm (all three phases) partially or fully in hardware. In that case, for example, instead of 100 extra PCs, one would need only 10 extra printed circuit boards to plug into the single Internet provider PC. The work of the authors was to device an algorithm of interest for a given R+D sponsor, and to implement it on specific universal boards based on the Xilinx FPGA chips.

In order to anticipate usefulness of a web page, proposed solution keeps track of users actions on a web page. A modified web client informs a dedicated server of users' actions performed on a web page, such as printing, bookmarking or mailing. Latter search algorithm relies on these information when ranking pages according to their usefulness.

The Obelix system collects the input from users; it gathers information about large variety of events and rank pages according to the assigned events. Thus, a page that has been visited thousand times will get higher rank then a page that has never been visited, even if the second one has more keyword matches than the first one. The goal of the Obelix system is to collect the access information and to provide it to the search server in the appropriate form. By performing these tasks, an Obelix server needs to process a large amount of data. In order to conduct a real-time data processing and acquisition, reconfigurable architecture is introduced.

2.1. Proposed solution

The Obelix search engine uses slightly modified web client-server approach. Namely, web browser is modified to inform Obelix server(s) about client actions trough the datagram connection. A connectionless protocol was chosen due to a high information workload. A loss of a single packet is of a minor importance to the search server, while a connectionless protocol reduces communication overhead. Each packet created by web browser contains information about the URL, and the performed action (saving, printing, mailing…) with its score.

When a user makes a search request, the search engine ranks web pages according to the number of the searched words in the page (or some more sophisticated conventional algorithm), and a Casselman score, which represents the sum of overall scores of user actions. These scores are combined into a general score according to the following formula:

$$S = S_f \cdot \sum_{DB} S_c$$

where S is general score, S_f is the frequency score, and S_c is the Casselman score.

The Obelix engine receives and processes the packets. In addition to the search server engine, like Altavista or Lycos, the search server site includes also the Obelix server collecting information about users' actions; it consists of a machine that accepts PCI cards (e.g., a PC or an Alpha) with a number of VCC Hot II boards (based on Xilinx 4000 or Spartan) plugged into the PCI slots. The number of VCC Hot II boards depends on the system capacity and traffic load, and could vary in a relatively large range.

The information's acquisition process consists of three phases: packets' collection, calculating sums, and transferring results to the database. Every phase is implemented as a separate HOT II configuration.

2.1.1. Information collecting phase

Modified web browser pack information into an IP packet of the following form:

Packed ID	Number of actions in the package
Packed URL	Performed action (1 byte)
…	
Packed URL	Performed action (1 byte)

The URLs are packed with static Hoffman compression. This way, there is no need to ever unpack URLs on the server side, since the same URLs from different clients have the same code. The actual codes are predefined on a large statistical sample of URLs.

Users' actions are defined to cover most of the users' operations with a web page. Actions included in the present version of server are:

- Visiting a page
- Saving a whole page
- Saving a part of a page (image or other object)
- Printing
- Following links from a page
- Cut/copy
- Bookmarking
- Custom dialog for page ranking

- Time spent visiting a page

After being received by server's network card, a packet travels PCI bus toward the appropriate HOT II board. A board receives a package and stores it into its memory for latter processing.

There are three important issues to consider while designing collecting phase: load balancing, detecting fake data, and fault-tolerance.

If several boards are present in the system, one has to design the policy of assigning packets to boards. One possible set of policy is based on sender's IP number, while other one is server-controlled distribution to idle board. While the first solution suffers of loss of packets if a destination board is in the processing phase and thus it cannot accept any incoming packets; the second one minimizes number of lost packages but generates higher traffic on the PCI bus.

Serving in a real world, the Obelix system is open to different misuses. Web site owners might generate fake traffic to boost up their web site scores. The Obelix server includes detection and prevention of these cases. Each board includes event table in its memory, with a following structure:

Client IP addr.	No. of req.	Last arrival time

Each client is allowed to send a certain number of actions during a defined period of time; others are ignored. Last arrival time field is updated whenever a new action comes, with arrival time more than defined period of time latter then Last arrival time. Another important issue is the possible fault-tolerance of the system. Namely, if the appropriate or all HOT II boards are busy calculating sums, then a packet that is arriving trough PCI bus gets lost. Regarding this fact, one can define two policies, one that permits and the other that doesn't permit loss of packages. In case of systems with high workload, possible loss of some portion of user information is acceptable, and sometimes even preferable if reducing communication load. Nevertheless, a fault-tolerant algorithm was developed for systems needing it. A fault-tolerant system stores packets into the main memory apart from storing it into the boards, into an entry like following:

Ack	Arrival time	Received IP package

If a board collects the package from a PCI bus, it sends package ID to the CPU, setting Ack bit to one. Package ID is a 16-bit word randomly defined by client. It is not unique in any sense, rather random enough to reduce possible collision and loss of packages. After some time CPU goes trough the table and processes all non-acknowledged entries, deleting others.

2.1.2. Calculating weighted sums

When a board receives a certain number of packages, it starts processing it. The purpose of this processing is to make weighted sum of all actions related to a certain URL, where each action has its own weight according to its importance (printing is more important then visiting, etc.) Naturally, not all occurrences of an URL will appear on the same board. Therefore, extra processing in the database system is needed, but its time is greatly reduced by calculating these partial sums in HOT boards.

There are several basic units in a board's Xilinx chip. Each one has the architecture as shown below:

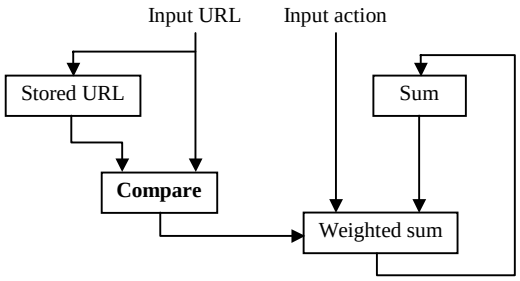

Figure 1 - Structure of a basic processing unit

The basic units are connected into a daisy chain (Figure 2). A unit compares the stored URL with a URL on the input. If they are the same, the unit updates the weighted sum, and invalidates the input into the HOT II board's memory. Otherwise, it forwards the package to the next unit. If there is no URL stored into the unit, the unit stores the input, updates the sum and invalidates the data into the memory. In this case, the speedup depends of the number of the basic units. After finishing the pass, it makes a new entry into the memory with the URL and a calculated sum. This record is latter forwarded to the main memory and to the database system.

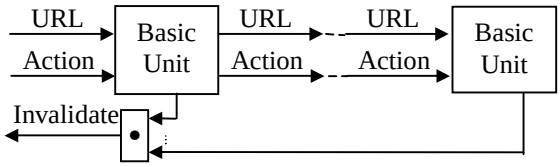

Figure 2 – A daisy chain of basic units

2.2. Search Results

Computer architecture, based on reconfigurable paradigm has been used for system implementation.

The architectural simulator of the system is implemented in order to test price/performance rate of the system.

The performance simulation of the system has showed the improvement of the search results using the presented techniques. Improvements vary, and go up to 10% for some subject categories. During the simulation, several other performance obstacles have been spotted, and will be addressed in the future work. Among these obstacles are: flexible string matching, keyword-related database, distinction between index documents and detailed documents. Nevertheless, the Obelix has proved itself as a valuable search engine offering more useful search results than conventional search engines.

3. Auterix: Home automation on the Internet

One attractive feature being offered by the Internet infrastructure is to be able, from a remote site, via Internet, to control processes at homes, in offices, or at production facilities. One possible scenario would be that a set of sensors collects the necessary information at a remote site and (using the ordinary telephone line) updates a central database that is accessible via Internet. These sensors can be related to home automation, in which case sensors collect data about the usage of utilities (electricity, gas, water, sewage, etc.) or resources (alarm, washer-drier, video-recorder, an iron, etc...). A somewhat different set of sensors could accommodate a remote office, or a remote factory (with no people and no internal lights, since the light from battery lamps can be used only during the periodical "greasing" of the production equipment).

For a given home, office, or factory environment, the most complex engineering tasks are: (a) to design the architecture of the system, (b) to implement a chip which enables periodical data transmission over the ordinary phone line, no matter if the line is busy or not, and (c) to implement the central database software. Possible benefits of such an environment are many. For example, in the case of utilities, with the related infrastructure installed, a utility company can offer the following deal to its customers (highly attractive): Your utility spending can be measured once every hour or even more frequently, which means that abnormalities like leakage can be detected promptly. In such conditions, each customer can get an insurance contract that authorizes the utility company to send up a repair crew as soon as something is noticed, with the first so many dollars of the repair cost not to be charged to the customer. Utility provider with such an attractive offer will attract a relatively large number of customers, and consequently the cost of the insurance will drop. Once it drops enough, the utility provider may decide to offer it for free, just to attract even more customers. In conclusion, here is one possible scenario of interest for customers: You come back home from skiing and you find a note saying that your water pipe broke in the meantime, that the leakage was immediately noticed, and that the repair crew was sent out at once, to repair it at no charge to yourself. An alternative scenario is that you return back home and find your basement full of water.

One possible scenario for the Internet automation implies the architecture depicted in Figure 3. The heart of the system is a specialized IASP (Internet Automation Service Provider). It has two interfaces: (a) One towards the service customers (homes, offices, production facilities, etc.) not connected with the Internet, based on some kind of the appropriate communications infrastructure; (b) One towards the Internet, and its remote users connected into the system via static or dynamic means (TCP/IP or TCP/mobileIP).

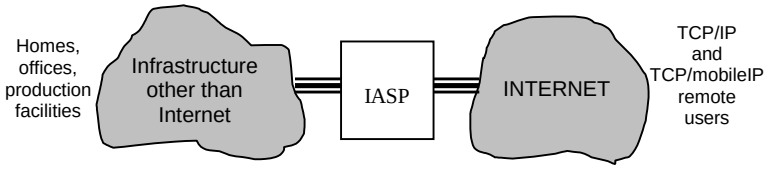

Figure 3 – The Internet Automation Architecture

In general case, service customers are equipped with measurement units (including the meters for electricity, water, sewage, etc.) and also with remote control units (including the remotely controlled robots). Both measurement (status collection) and control (action reinforcing) are organized around a specialized microcontroller located in remote homes, offices, or production facilities. On one hand, these microcontrollers are connected with meters and robots. On the other hand, these microcontrollers are connected with the communications infrastructure, linking them with the IASP.

In general case, among the Internet users, some are equipped with meters (in the widest sense) that can be read by accessing the corresponding Web site, and/or robots (in the widest sense) that can be manipulated by accessing the corresponding Web site, and clicking to appropriate commands/controls. Some of the service customers are, at the same time, also remote Internet users. The working scenario is based on the following (or a similar) procedure.

The IASP periodically collects the measurement/metering information from measurement units of service customers (remote home, office, or factory data), stores it in its memory, reformats it, and makes it available at an appropriate Web site, for interested (and authorized) parties to access it.

After an interested party accesses the information of interest, the appropriate decision-making process is performed, and certain actions are requested, via the service customer interface (which implies a direct access) or the Internet user interface (which implies an indirect access). Consequently, some switch is activated or deactivated, or some robot is made to act or to stop acting in the desired way.

The described system enables a number of important business scenarios. They are related to home automation, office automation, and operation of remote factories.

The entire described architecture and the corresponding procedure have one important bottleneck. On the interface towards the service customer, the question is what communications infrastructure to use; basic telephony infrastructure seems to be the obvious first choice, but also the natural source of the major system bottleneck.

Customers, utility companies, business and production facility owners can benefit from Internet automation architecture. It founds the basis for improvement of all technical, service quality, and economical aspects of home and business automation. Special care should be taken when implementing the service customers communication part of this architecture, since it can be the source of a major system bottleneck. The careful achievement of performance/complexity tradeoffs leads to a design with the optimal performance/complexity ratio, which may be of the greatest importance if custom chip or reconfigurable FPGA VLSI implementation of the design is the primary intention.

4. Netrix: Modifying Netscape for better business opportunities

Now, Netscape is in the public domain, and users are allowed to modify it and to update it. This opens up a number of different new research avenues leading to new applications. In order to organize experiments to learn about typical customer behavior in the context of the global Internet market, one can modify Netscape, to collect statistics of Interest. Changes can be implemented as patches or plug-ins. The modified Netscape is first used to detect a specific behavior, and then to utilize it in a specific business environment, with the major goal of maximizing the revenue.

5. Mobix & Proxix: Intelligent and efficient genetic search and proxy caching with mobile agents

Search using the systems like Altavista or Yahoo is based on indexed search. This means that an item cannot be located before it is indexed. Indexing may take days, or weeks, or even more. However, for some type of e-business on the Internet, it is important to be able to locate the item before it is actually indexed. For example, this is important for those who may like to be able to access a new product (with a better price/performance ratio), before it is noticed by competition, so they can maximize their profit. The solution in such cases is to use search methods based on links (like genetic search or simulated annealing). Links based search methods assume that (in well established markets) interest groups and industry consortia reference one another on the Internet, and that there exists a fairly large probability that starting from one WWW presentation, by following the links, one can arrive to the desired new WWW presentation that is not yet visible on indexing based systems like Altavista or Yahoo. One starts from a key WWW presentation that includes URL links to other WWW presentations. Once these other presentations are downloaded, their benefit factor is computed, and the best ones are selected for further processing. The URL links from these new presentations are used to download additional WWW presentations, and the process continues. Periodically, the selected URL pointers are modified for better price performance. This process is called mutation. Typically, mutation is based on mutational databases. The authors have experimented a lot with mutation algorithms that exploit the principles of spatial and temporal mutation (as they can help considerably in the not so well organized markets). Finally, the system can benefit a lot from the utilization of mobile technology. Often times, a gigabyte has to be downloaded, in order to generate a binary information (to invest or not). This represents a waste of bandwidth, which costs time and money. An alternative approach would be to send out a mobile agent to the remote site, in which case only the binary result has to be downloaded from the network. Of course, if this is not possible, the static agents have to be used.

Intelligent proxy caching is of interest for Internet providers. The more the information is reused, the lower are the bandwidth requirements. Consequently, investments into a larger bandwidth can be postponed for a later time, when the price of investment will be lower. Efficiency of the system is further improved if some intelligence is added to figure out if the change done on a WWW site (since it was last cached) is in the part that is needed by the customer, or in another part of no current interest to the customer. If that is the case, the cached copy can be reused, although the original copy has been changed. Additional improvement comes with the application of mobile agents. In that case, as soon as a page is cached, a mobile agent is sent to the remote site from which the page was cashed, to monitor further changes. Of course, this is not possible if remote site does not welcome the foreign agents. Consequently, only a hybrid mobile/static solution is realistic.

5.1. Proposed solution

As the number of documents and servers on Internet grows with the enormous speed, it becomes necessary to design efficient algorithms and tools for search and retrieval of documents. Also, the number of accesses to servers on Internet constantly grows. Congestion of servers and links can be alleviated using proxy caches. Latency on Web can be reduced using prefetching and caching. Efficient search of documents can be done with improved genetic algorithm that exploits the principles of temporal and spatial locality.

Mobile agents can be used to optimize network traffic for distributed applications.

Improved genetic algorithm includes mutations exploiting spatial and temporal locality. The idea of spatial locality exploitation is to examine documents in the neighborhood of the best ranked documents so far, i.e. the same server or local network. Temporal locality concerns maintaining information about previous search results and performing mutation by picking URLs from that set.

This solution is the best present solution in static domain, but a lot of time is spent for fetching documents from the Internet onto the local disk, because content examination and evaluation must be performed off-line. Thus, a huge amount of data is transferred through the network in vain, because only a small percent of fetched documents will turn out to be useful.

Proxy caches are located between users and Web servers. They store most popular pages, which are delivered to the user on their request. This reduces network congestion, long response times, and reduces a number of hits to popular Web servers. But, delivered copy of the document can be stale. To prevent this, Web servers are asked from time to time if change has happened since last retrieval. But these "questions" travel over long distances, put additional load to the network and reduce the accuracy of caching.

The logical solution of these problems is to construct mobile agents that would transfer themselves onto the home servers and examine documents at remote sites, transferring back only the needed documents and data. In addition to lower data transfer over the net, great improvement is achieved using parallel data processing. In genetic algorithm, multiple independent subpopulations each run a genetic algorithm on their own subpopulations and periodically fit strings migrate between the subpopulations. This approach achieves significant gain in time and disk space, especially in the module that calculates Jaccard's score since only these numbers is sent back through the network, instead of all documents that are potential candidates for the result set. To improve proxy caching mobile agents can pole Web server for changes and inform proxy if some page has been changed. Since requests are issued closer to the server their frequency can be increased without overloading the net.

In our experiments in genetic search and proxy caching, *Concordia System* and tools were used. A *Concordia System* consists of multiple machines in a local or wide area network, each of which is running a *Concordia Server*. The Concordia Server is responsible for providing all Concordia functionality on a given machine, including the basic agent mobility and remote administration. Each Concordia Server includes a number of *Services*, which provide specific functionality on the machine.

To enable the mobile agent system, it is necessary to install Concordia Server on every machine within a network. It enables agent transfer and interface between the visiting agents and the host system.

Concordia includes a collaboration framework that enables multiple agents to work together and coordinate their actions. Agents within an application may form one or more collaboration units.

Agents could use several communication technologies such as CORBA (Java/IDL), sockets, or Concordia Distributed Events for network communication.

In Concordia there is a distributed events framework that enables agents to communicate with each other either synchronously or asynchronously. They are extremely useful for notifying objects of changes to resources and unexpected conditions.

RMI is another mechanism by which an agent can interact with other objects on a network.

In a distributed application some of the implementations are assumed to reside in different virtual machine. One of the central and unique features of the RMI is its ability to download code of an object's class if the class is not defined in the receiver's virtual machine. The types and the behavior of an object, previously available only in a single virtual machine, can be transmitted to another, possibly remote, virtual machine. This allows new types to be introduced into a remote virtual machine, thus extending the behavior of an application dynamically. This ability is enabled by the dynamic nature of the Java platform, which is extended to the network by RMI. RMI dynamically loads code into the receiver's Java virtual machine and runs the code without prior knowledge of the class. Such applications that have the ability to download code dynamically with RMI are part of the basic mechanisms for distributed programming on the Java platform.

To make our mobile system more tool-independent and to improve communication, system is upgraded with *Monitor* module. This module is responsible for launching and monitoring mobile agent modules, as well as for managing communication through the network among these modules.

The idea is to make communication through the net and managing mobile agent system transparent to the applications modules. The applications are organized with one static *Server Application* on the local server and several *Mobile Application* modules on remote servers for executing tasks. These modules communicate through *Monitor* and corresponding *Client Agent* module on remote server using standardized message format for sending and act like static application modules located on the same server.

The *Monitor* module after receiving a message from the *Server Application*, analyze the content and determines the *Client Agent's* target location. For the reason that the *Server Application* do not know exactly where *Mobile Application* modules are located, it is possible that there is no *Mobile*

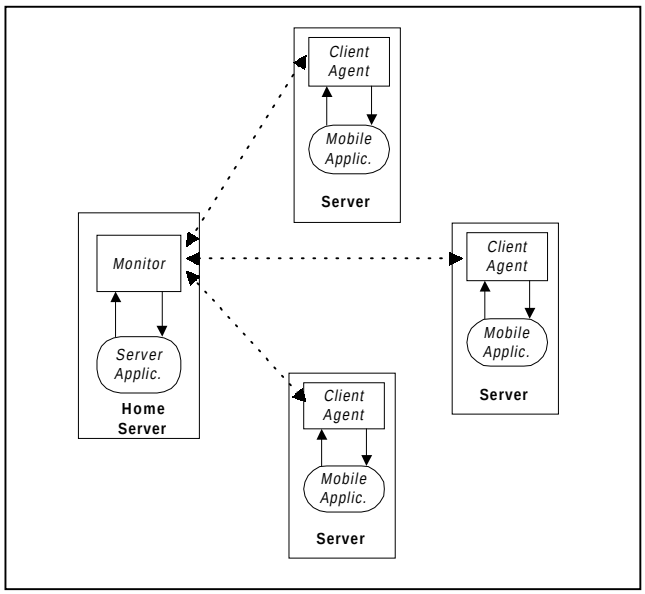

Figure 4: Communication through *Monitor* module. *Server Application* on local server and *Mobile Applications* on remote servers send messages through *Monitor* and *Client Agents* modules.

Application module running on remote location or even *Client Agent* module. In that case *Monitor* sends message to the *Client Agent* on the remote server to start new *Mobile Application* module and to pass the message, or first sends *Client Agent* to the new location to bring the *Mobile Application* code to the remote location and to start it there.

To reduce necessary transfer of *Client Agents* several improvements are implemented:

- after evaluation of the document *Client Agent* stays on the new location and waits for further instructions. In the case that evaluation of new document on the same server is needed (e.g., because of spatial locality), the *Monitor* only sends a message to the *Client Agent* on the server to evaluate the document, instead of sending new *Client Agent*. To avoid mistakes, every *Client Agent* has its ID and keeps the ID of the current search. In intention to free the resources of the host server, *Client Agent* dispose itself after given time of inactivity.

- the *Client Agent*, after sending results to the *Monitor* can send its clones to the new locations following hyperlinks in the documents.

- the sending of *Client Agent* can be done from the remote location where *Client Agent* is already sent. That is, the *Monitor* can send a message to the *Client Agent* on the remote location to send itself to the new locations. That reduces data transfer on the home server.

In a present time, there are no many Concordia servers installed on the Internet, so the possibilities to send mobile agents are reduced, but improvements in performance are still significant. *Client Agents* are sent to the nearest location on the Web and with trusted server and the *Mobile Applications* can fetch documents for less time and less net transfer. In this solution several *Mobile Applications* work in parallel instead of only one application installed on the local server.

For experimental use, *Monitor* module has the table of trusted servers on the net, where it can send its mobile agents. Every time when new location occurs for sending agent, *Monitor* checks the location if there is Concordia server installed and if it can send an agent to that location. If agent launching succeeds in a specified *timeout* period, *Monitor* will update its table of available servers adding new data. This table is stored for later use.

For the reason of improving performance, *Client Agent* modules are remaining on remote locations as long as they have *Mobile Applications* to serve or until given time of inactivity. Also, several *Server Applications* can use the same *Monitor* module on the local host and *Mobile Applications* are using the corresponding *Client Agent* module available to more than one of these modules on each remote host. The first time when *Server Application* tries to send the message through *Monitor* module, *Monitor* registers the new application to the local table of applications. This table is used for local communication among *Monitor* and *Client Agent* modules and applications do not need to store their ID. When *Server Application* is finishing its work, it sends a message to *Monitor* module to clean up corresponding *Mobile Applications*. Then *Monitor* sends message to all *Client Agents* to terminate corresponding *Mobile Applications* and it will release resources on the remote server.

The software agent system is modular that makes it easy for the further improvements and for upgrading with new application modules. These new applications would be developed as any other static application using the existing infrastructure for communication and enabling mobility without the need to use some other tools.

5.2. Search Results

The mobile implementation shows significantly lower transfer over the net. This is because in static implementation all the data for evaluation are sent over the network, which can be significant versus mobile implementation where only several *MAgent* “clones” are sent. For the same reason, the quantity of transferred data in static implementation grows much faster with number of output data.

The direct implication of less transfer overhead is a less run time needed. In mobile implementation evaluation of the documents are done in parallel on remote servers and only the results are sent to the home server. All that makes a significant improvement in performance specially when more output documents are needed.

Current caches are usually set to check freshness of locally stored copy of the document once per day. Adaptive algorithm in the worst case checks twice per day but won't miss more than 12 hour more than once. So the accuracy is increased at least twice.

Mobility reduced network load since poles are performed on shorter distances (usually measured in hops). Web server is not overloaded since poling intervals depend on dynamic of change of each document. There is no exact calculation to prove that since it depends on document's history. But the worst case is when document doesn't changes at all and mobile agent is delegated for only one URL. Standard proxy cache will check once per day while mobile agent will check twice. If distance between proxy and Web server is more than twice the distance between mobile agent and Web server, saving is obvious.

Presented infrastructure forms bases for additional improvements of search and caching service. Next step is exchange of information between caching and searching module. The URLs of the most popular pages found in the cache could be submitted to searching module as a candidate for a URL in the database. It is to expect that those popular documents could be the target of someone's search, especially in environments, where group of people share interests in some subject (*e.g.* companies, departments...). Also, it is to expect that highly ranked URLs in a search result will be fetched. To reduce user's fetching time**,** those documents could be prefetched. End users will have all desired information as soon as possible and as accurate as possible.

6. Hashix: A search chip for ATM based on hashing

The ATM allows for cells to be lost at the physical transmission layer, and includes no automatic retransmission on the level of the physical transmission layer. For transmission of video on demand or image in general, this is not an issue (if one pixel on a motion video screen is lost, no human eye will notice it). For transmission of bank data, this is a serious issue, but the system can request retransmission on the higher software levels. One of the most crucial aspects of the ATM routing design is real-time search (within one ATM cell interval), through unsorted tables that could be of an extremely large capacity (10000 entries or more). For search without loss (i.e., which always performs the task within one cell interval) one needs complex parallel search engines. However, since ATM tolerates loss of cells (one in 10^6 to one in 10^{10}), a simple hashing based method can be utilized that trades off memory for the complexity of the search engine. Consequently, the authors were able to design and implement a search chip that is orders of magnitude less complex.

7. Reflectix: An FT I/O pump based on reflective memory

Disk I/O intensive applications often times require all data to be stored on two (or more) different discs, for higher reliability and fault tolerance (FT). In the case of applications with a high degree of sharing, replication of data is needed for faster access by the end users. Reflective memory represents an efficient solution in both cases. Data written to memory of one PC in the system automatically gets reflected into the memory of other PCs in the system. If it is shared data, reading will imply no consistency maintenance related overhead. If it is data needing high levels of protection, it will be concurrently stored into two or more disk units. The authors have designed and implemented a board that enables all the described scenarios to be utilized in the PC environments.

8. Multimedix: The next generation PC for multimedia applications

Efficient processing of multimedia applications requires PCs based on multiprocessing, enhanced with a variety of different accelerators for time consuming system or user processes. Architecture of the system, caching strategies, and design of specific accelerators are of crucial importance for efficient system implementation. The authors have worked on all three aspects of the problem, and concrete designs (ready for production) have been generated.

9. Siliconix: Modeling for silicon compilation

The authors have designed several models of processor chips or processor accelerators for general purpose or special purpose applications in multimedia and e-business. The typical working scenario is as follows: (a) Conceptualization of the architecture, (b) Simulation analysis and testing on the functional level, (c) Creation of the production mask. Major designs include a 200MHz 32-bit RISC microprocessor for DARPA (about a decade before Intel) and a clone of an Intel's 64-bit RISC microprocessor for a Japanese customer. Both efforts are fully described in the open literature.

10. Gipix: Multiprocessing for GPS integrated into the Internet

An ongoing project examining the best multiprocessor architecture for GPS (global positioning system) in the context of maximizing the system exploitation using the Internet infrastructure.

11. Aquilix: A software package for conference organization

Collaboration and workflow on the Internet are topics of special importance. One of the issues is organizing the Internet-based conferences. An appropriate software package was developed and tested at the SSGRR-2000.

12. Socratenon: A software package for education on the Internet

A project was aimed at designing the complete software infrastructure for electronic education on the Internet. This infrastructure is more widely applicable, and could be utilized in related fields.

The goal was to design, implement, and study exploitation outcomes of a Web-based training system. It was envisioned to be a user-friendly, easy maintainable intelligent educational environment. First, several design details had to be considered. Primary issues of such a design are data warehousing, distribution and multifaceted manipulation on one hand, and intelligence incorporation on the other. It was decided to rely on techniques that were broadly applicable, as well as standards and packaged solutions that offer required functionality. For instance, we needed a platform that supports abstract objects and can easily deploy them to a thin-client (any regular Web browser without additional software or hardware needed) in a tangible form. Also, underlying database management should allow easy access to these objects and allow easy functional upgrades.

The environment was meant for wider community, and objectives were improvements in training and learning effectiveness, reduction the training costs, more sophisticated industrial intellectual capital retention, declined learner training time, and steeper learning curve.

Additional requirements included need for dynamical customization to a wider spectrum of client devices (PDAs, mobile devices, etc.) connected to a network (Internet, Intranets, wireless LANs, etc.).

The outcome of the project includes the basic platform, customizable to various users and domains of knowledge, together with well-established methodologies and qualified competencies tested on the pilot-case.

12.1. Proposed Solution

In traditional training environments it is customary for students to pose questions to their tutor, when some uncertainty needs clarification. Socratenon offers means of communication between students and teacher on several levels (from chat and forum to ICQ look-alike; VRML interface is under development). This feature would enable synergistic effect that is the main quality of classical approach.

Learning process is learner-centered, so that learning material is tailored to every particular user. Socratenon makes use of structured knowledge (rich with metadata), student model (sketch of student made during student learning sessions by monitoring student's behavior), and expert model (describing learning strategies). Using these structures, Socratenon can perform curriculum customization, deciding on how knowledge is sequenced, how the material is presented, and when and how hints should be offered or organized. This means that every student gets his/her own, individualized course. It's just like having a well-informed personal tutor on fingertips.

Special attention was given to user interface (UI) so that everything needed for specific learning session occupies not more than three screens, avoiding confusion of too many windows. Hints for the highlighted items are available on a single mouse-click.

It is important to achieve the most comfortable learning process. This is provided by imaginative blend of UI design and interface programming, relying on robust interoperable technologies such as eXtensible Markup Language (XML), eXtensible Style Language (XSL), Cascading Style Sheets (CCS) and Dynamic HTML, supported by Java programming language.

Web delivery solution has all of the advantages users have come to rely on from a web browser (ease of wide distribution with central update, friendly/familiar user environment), while providing advanced functionality through comprehensive Java development. It uses the W3C Document Object Model (DOM) Recommendation to control behavior associated with each element in a document; rendering of the elements is specified using the latest W3C Draft for the XSL.

This approach enables development of innovative environment that would incorporate tangible benefits of both virtual and traditional learning environments, while minimizing the shortcomings of both approaches.

Choosing the right blend of good practices in both traditional and existing on-line training environments was a key to success of Socratenon training environment.

Examination of learning and teaching tendencies in both environments has led to creation of a hybrid Web-based training concept, open to future developments and insights in technology, pedagogy and Internet domains.

Making several design decisions was crucial to make the system modular, extensible and above all, useful to both users and future developers of Socratenon. We are looking forward to new creative input from inside and outside Socratenon environment.

The existing version has been used by industry and academia. An interesting application is related to teaching/learning of Italian language (*http://desert.etf.bg.ac.yu:8080/*).

Future plans for Socratenon include proving the concept by utilization in a few diverse fields of study. More research & incorporation of adaptable AI techniques is expected to follow. Immersive environments will surely win the battle over the ultimate user interface, so integration of VRML-based interface is also being planned.

Conclusion

Research and development presented here cover the activities of the authors during the decade of 90's. Details can be found at the WWW (http://rti7020.etf.bg.ac.yu/rti/ebi and http://galeb.etf.bg.ac.yu/~vm/), or in selected books listed below.

REFERENCES

Milutinovic, V., *Surviving the Design of a 200 MHz RISC Microprocessor: Lessons Learned*, IEEE CS PRESS, 1997.

Milutinovic, V., *Surviving the Design of Microprocessor and Multimicroprocessor Systems: Lessons Learned*, WILEY, 2000.

Milutinovic, V., *Infrastructure for E-Business on the Internet*, Publishers Still Bidding, 2001 (http://galeb.etf.bg.ac.yu/~vm/books/2001/ebi.html).

Cvetkovic, D., Petkovic, D., Pesic, M., Horvat, D., Milutinovic, V. "Architecture of the Mobile Environment for Intelligent Genetic Search and Proxy Caching," *Published in the Proceedings of the HICSS-33, held in Maui, Hawai'i, USA. January 4-7, 2000*

Horvat, D., Cvetkovic, D., Milutinovic, V. "Mobile Agents and Java Mobile Agents Toolkits," *Published in the Proceedings of the HICSS-33, held in Maui, Hawai'i, USA. January 4-7, 2000.*

Nikolic, N., Trajkovic, M., Milicevic, M., Milutinovic, V., De Santo, M. "Socratenon - a Web-based Training System with an Intellect," *Published in the Proceedings of the HICSS-33, held in Maui, Hawai'i, USA. January 4-7, 2000.*

Knezevic, P., Radunovic, B., Nikolic, N., Jovanovic, T., Milanov, D., Nikolic, M., Milutinovic, V., Casselman, S., Schewel, J. "The Architecture of the Obelix - An Improved Internet Search Engine, " *Published in the Proceedings of the HICSS-33, held in Maui, Hawai'i, USA. January 4-7, 2000.*