Multi-agent Optimized Load Balancing Using Spanning Tree for Mobile Services

Dr. Pradeep Kumar Sinha, Sunil R Dhore

ABSTRACT

Various communication and computing tasks in the fields can be integrated and applied in a distributed system. However, those resources are heterogeneous and dynamic in nature, connecting a broad range of resources. This study proposed a hybrid load balancing policy to maintain performance and stability of distributed system in Mobile services. Proposed work suggests to opt the use of some of advanced and efficient technologies like Multiagent. In this proposed implementation two models are developed. The worker model is used to describe the workload and its current distribution within the system. The master model defines for a given algorithm at any instant of time and shows the formal context for obtaining and evaluating the load distribution decisions. Multiagent computing on a cluster of workstations is widely envisioned to be a powerful paradigm for building useful distributed applications. The Mobile agents of the system span across all the machines of a cluster. Just like the case of traditional distributed systems. With different characteristics between ordinary processes and agents, it is interesting and useful to investigate whether conventional load-balancing strategies are also applicable and sufficient to cope with the newly emerging needs, such as coping with temporally continuous agents, devising a performance metric for multi agent systems, and taking into account the vast amount of communication and interaction among agent. This work discusses the above issues with reference to agent properties and load balancing techniques and outlines the space of load-balancing design choices in the arena of multi agent computing. The proposed algorithm works by associating a credit value with each agent. The credit of an agent depends on its affinity to a machine, its current workload, its communication behavior, and mobility. When a load imbalance occurs, the credits of all agents are examined and an agent with a lower credit value is migrated to relatively lightly loaded machine in the system. Proposed work considers the problem of load balancing to minimize the cost of dynamic computations, including the cost of migrations. We propose the Ripple load balancing paradigm, the load balancing service presented is a generic tool for enhancing performance of accessing distributed objects from the WAP interface.

Index Terms

Load balancing, Multiagent, WAP

INTRODUCTION

With the growth of wireless subscribers' demands, service providers and operators face the challenge of maximizing their network capabilities with their existing infrastructure. This paper develops our framework to optimize the use of network resources to answer subscribers' demands and to reconfigure services deployment on network when necessary. To achieve the same minimum conditions is that a load balancing algorithm should meet are stability, the load eventually reaches a fixed distribution, and levelness, the load at the processors is equal at the fixed distribution.

Multiagent systems have recently been widely employed in developing scalable software system on heterogeneous networks. Indeed, using a cross-platform language, distributed systems based on agents are very attractive because of the inherent scalability and autonomy [3]. Viewing the software agents as "brokers" and interactions among agents as the exchange of "services", a multiagent closely resembles a community of human beings doing business with each other, and is widely envisioned to be able to perform many commercial activities on behalf of human being.

First motivation behind this work is given (section II).we will study the classification of load balancing algorithm (section III). Then, we present the concept of multiagents and mobile agents and their use in load balancing (Section IV). We introduce the ripple load balancing, which has several advantages (Section V). Web architecture with its interface to WAP and objectives are discussed in Section VI.

A. Motivation

Locally distributed system consists of a collection of autonomous computers connected by local area network. Users submit tasks at their host computers for processing. The need for load distributing arises in such environments because, due to the random arrival of tasks and their random CPU service time requirements there good possibility that several computers are heavily loaded (hence suffering from performance degradation) while others are idle or lightly loaded.

Clearly if the workload at some computers is typically heavier than that at others or if some processors execute tasks at slower rate than others, this situation is likely to occur often. The usefulness of load distributing is of as obvious in system in which all processors are equally powerful and over the long term, have equally heavy workload.

Statistical fluctuation in arrival of tasks and tasks services time requirements at computers lead to high probability that at least one computer is idle while a task is waiting for service elsewhere. Their analysis can be presented as model of computer in a distributed system by M/M/1 server.

Consider a system of N identical and independent M/M/1 servers. By identical we mean that all servers have the same task arrival and service rates.

Let,
ρ = be the utilization of each server
Then
P0 = 1 - ρ is probability that server is idle

Let,
P = be probability that the system is in a state in which at least one task is waiting for service and at least one server is idle.

Then

$$P = \sum_{i=1}^{N} \binom{N}{i} Q_i H_{N-i}$$

Where: Qi= probability that given a set of i servers are idle.
And H_{N-I}= is the probability that given set (N-I) are not idle

H_{N-1} = {probability that (N-i) systems have at least one task} – {probability that all (N-i) systems have exactly one task}.
Therefore:
$P = 1 – (1-Po)^N (1-Po^N) - Po^N (2-Po^N)$

We can find out the values of P for various values of server utilization ρ and number of server N

1. For moderate system utilization (ρ =0.5 to 0.8) the value of P is High, indicating good potential for performance improvement through load distribution.
2. At high system utilization the value of P is low as more servers are likely to busy, which indicates lower potential for load distribution.
3. At low system utilization the value P is low as server are idle

Another observation is that as the number of servers in system increases, P remains high even at high system utilization.

B. Classification of load balancing algorithms

C. Anatomy of load balancing:

Resources are distributed in different geographic locations. Stability and performance of each resource is different. In other words, newly distributed system is dynamic and resources are composed of heterogeneous resources. Thus, an important problem is resources selection and task distribution when task are executed. This study proposed a hybrid load balancing policy, which selects effective node sets in the stage of static load balancing to lower the odds of selecting ineffective nodes and makes use of the stage of dynamic load balancing. When a node status changes, a new substitute can be located in the shortest time to maintain the execution performance of the system [1]
It has four components:

1. Transfer policy that determines whether a node is in a suitable state to participate in a task transfer
2. Selection policy that determines which task should be transferred.
3. Location policy that determines to which node a task selected for transfer should be sent.
4. Information policy which is responsible for triggering the collection of system state information.

A transfer policy typically requires information on the local nodes state to make decision. A location policy, on the other hand is likely to require information on the state of remote nodes to make decisions.

D. Classification of load balancing algorithms

The allocation of workload in distributed systems has almost as many views as one would like. The attempt to find suitable structures to classify the different ways of solving this task can therefore reflect the underlying model only. We can classify these algorithms based on three models [2]: The load model is used to describe the workload and its current distribution within the system. The action model defines for a given algorithm at any instant of time the eligible next step(s). And finally, the solution model shows the formal context for obtaining and evaluating the load distribution decisions. All three models and their interrelationship can be compared with existing load distribution approaches. The result of the investigation is the recommendation that load distribution algorithms can be classified according the five criteria: objectives, type and amount of used information, the source of the distribution, the parameter time, and the initiating instance.

E. Online distributed Multiagent computing

Multiagent systems have recently been widely employed in developing scalable software systems on heterogeneous networks. Indeed, using a cross platform language (such as java in most cases), distributed system based on agents are very attractive because of the inherent scalability and autonomy. Viewing the software agents (usually in form of object) as “broker “ and the interactions among as exchange of ‘services’ a multiagent system closely resembles a community of human beings doing business with each other , and are widely envisioned to be able to perform many commercial activities on behalf of human beings.
Agent’s properties:

1. Reactive: responds in a timely fashion to changes in environment.
2. Autonomous: exercises control over its own actions.
3. Goal oriented / Proactive: does not simply act in response to the environment.
4. Temporally continuous: is continually running process.
5. Communicative / socially able: communicate with other agents, perhaps including people.
6. Learning / Adaptive : changes in behavior based on its past experience

7. Mobile: able to transport itself from one machine to another.
8. Flexible : actions are not scripted
9. Cloning: duplicates itself to achieve better performance.
10. Character: believable personality and emotional state.

F. Credit based load balancing model

Credit based load balancing model, which aims at capturing some of the necessary agents' characteristics. It lets us analyze the dynamics of load balancing operation with respect multiagent system.

In dynamic load balancing schemes, the two most important policies are selection policy and location policy.

If selection policy is formulated carefully, the desired effects are that the agent selected to migrate will not make the overall situation worse by making the destination more overloaded than the source and cost of the migration will be compensated by the gain in performance. Likewise should the location policy be properly planned, the overall system workload will be more averaged after the migration.

The credit based load balancing model focuses on these two policies.

We assign a numerical value called credit to every agent. The credit indicates the tendency of the agent to remain undisturbed in case migration is under consideration. To an agent, the higher its credit, the its chance to stay at same machine, which is equivalent to saying that its chances to be selected and migration is lower.

The credit of an agent increases if the following ways :

1. Its workload decreases
2. It communicates with other agents also residing at the same machine or
3. It has a high affinity with locale machine. For example, it requires special type of processors, I/O devices , or large amount of data localized at the machine.

On the contrary, the credit of an agent decreases in the cases below:

1. Its workload increases
2. It communicates with other agents residing at other machines.
3. It has a high mobility, i.e. it can be migrated elsewhere very easily, or
4. It has just sent or received an agent's message which indicates that the agents will probably become busier in a short while.

As interaction and communications among the agents continue the credit of each agent's changes accordingly, such a credit can be used in the selection policy, where the agents with the lowest value identified and migrated.

The location policy first identifies which remote agent will perform the most communication with the agent to be migrated. The machine at which this remote agent resides is selected as destination machine.

G. The Comet Algorithm

We assume an application is composed of agent's executable on any of P machines of the cluster.

The structure of the application is modeled by the interdependence relationships among the agents. More specifically, we use an undirected graph to model the application structure. An undirected graph is an appropriate generic model because a multi agent application executes perpetually and produces results continuously in response to user queries.

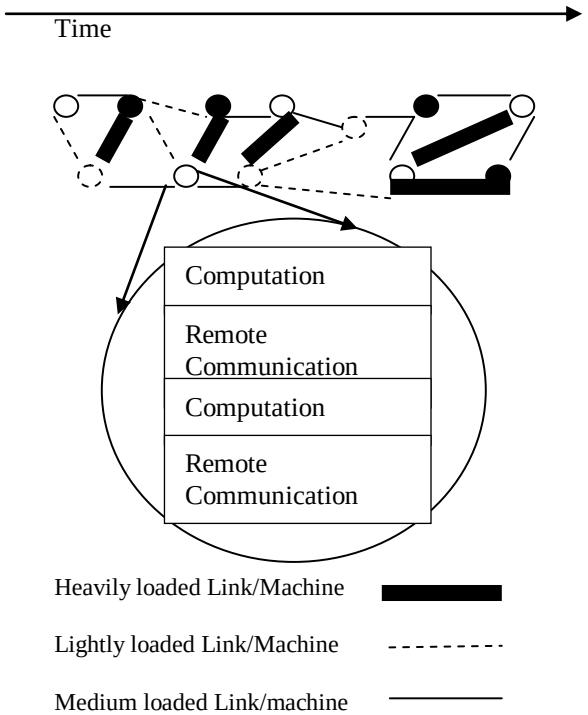

Fig 1: **Iterative and dynamic nature of a multi agent application and structure of agents**

Load of an agent executing on machine is defined as the sum of its computational load and communication load

$U_i = H_i + G_i$

Where :

H_i – Communication Load

G_i – Computational Load

The load L_k of machine m_k is defined as the sum of all its local agents load. More specifically

$L_k = \Sigma (w_i + u_i)$ where :

w_i – Communication Load
u_i – Computational Load

Goal of a load balancing algorithm is to minimize the variance of the load among all the machines in the cluster, this will turn minimize the average response time of serving users queries.

H. Load Balancing With Mobile Agents

The mobile agent is autonomous software unit, capable to move across the network and perform some defined actions on behalf of its owner. They migrate from node to node and return to its home node to report their results.

Mobile agents can be effectively used in many areas, for several reasons, including improvements in latency and bandwidth of client-server applications as well as reducing vulnerability to network disconnection. Mobile agents' following features are welcomed for load balancing and could be successfully utilized in increasing system performance for the following reasons:

1.They reduce the network load (instead of all-to-all communication, the agent visits all nodes in cyclic manner),

2. They encapsulate protocols (protocol for task exchange is encapsulated in the agent)

3.They execute asynchronously and autonomously (the agent's creator is free after agent's creation and dispatching),

14. They adapt dynamically (agent will react according to current situation)

5. They are naturally heterogeneous (thus, load balancing could be applied to heterogeneous systems)

6. They are robust and fault-tolerant (if a host or link is being shut down, all agents could change their paths and continue their operation on another host in the network).

During last several years, a few solutions for mobile agents based load balancing have been introduced. Some of them originated from behavioral patterns in the nature like ant's life and swarming intelligence. Their basic idea is to apply manners of communication and cooperation between animals to communication and cooperation between mobile agents[6].

I. Ripple based algorithms for migration and load balancing

We introduce a class of algorithms, called Ripple algorithms that reach leveling in time linear in the diameter of the processor graph. Ripple algorithms are based on the simple idea that if the load in the network is initially balanced then, any load increase (or decrease) in one processor should be equally distributed among all the processors. We begin by describing the Ripple paradigm for a linear processor array. In particular we present two algorithms; The first algorithm, Tortoise, minimizes migration cost, and the second algorithm, Hare, minimizes load imbalance cost [4].

The Ripple technique introduced here has many advantages; its time to stability is $O(d)$ where d is number of processors, it can be viewed as both sender initiated and receiver initiated [25], and its scheduling mechanism allows it to be very flexible.

J. Overview of proposed system

K. Proposed Load Balancing Systems Design:

All load balancing strategies involve the adjustment of the

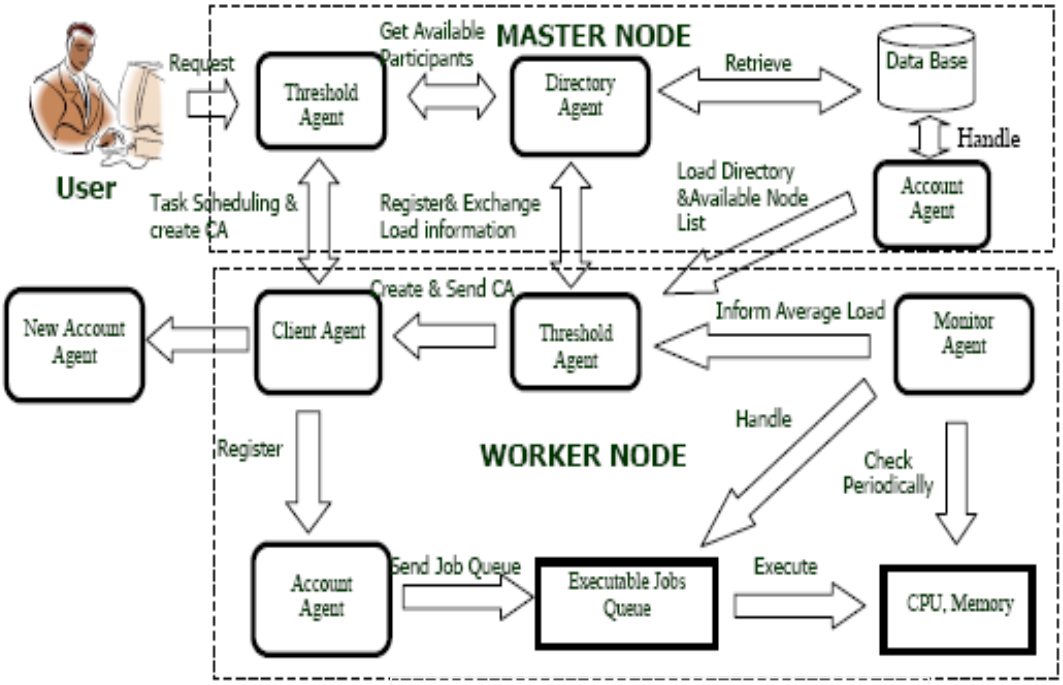

Fig 2: **Overview of Mobile Agent Based Load Balancing Environment**

distribution of the work load among the participating processors if the distribution is expected to result in a reduction of the total execution time. These potential reductions due to load balancing need to be weighed against the overhead involved in monitoring the progress of the computation, as well as the redistribution of work load among the processor.

The design of a load balancing mechanisms should include the following policies:

1. Information Gathering Policy: maintains the information about workload at the nodes in cluster. The policy is made up of two components: frequency of information exchange and the method for dissemination of the information. There is a tradeoff between having accurate information and minimizing the overhead. It also includes the estimation and specification of workload, e.g., processor load, length of queue, storage utility etc.
2. Initiation Policy: determines who initiates the process of load balancing. The initiator can be the source node, the destination node, or both (symmetric initiations).
3. Job Transfer Policy: decides when the initiator should consider reallocate the workload to be executed to other nodes. The decision can be made based on only local state or by exchanging global processor load information.
4. Selection Policy: determines which particular job to reallocate. Non _preemptive policies select tasks from the set of jobs, which are yet to begin execution. Preemptive policies expand this set to include all jobs located at the processor.
5. Location Policy: determines to which servers the jobs should be re-allocated. The simplest location is to choose a server at random. More complicated policies use negotiation, where the initiator negotiates with each member in cluster.

These policies must be represented and implemented in appropriate system components.

Our proposed architecture is based on multi-agent system. Mobile agents are software programs that can migrate from host to host in a network carrying code, data and state of the execution. They have the abilities to survive in disconnection network and reduce the network latency. Therefore, we use mobile agent in distribution of load in a cluster. They will serve as monitoring, controlling and distributing of load and keeping the directory service for the whole cluster nodes information.

L. Load Balancing Approaches for General Web Architecture

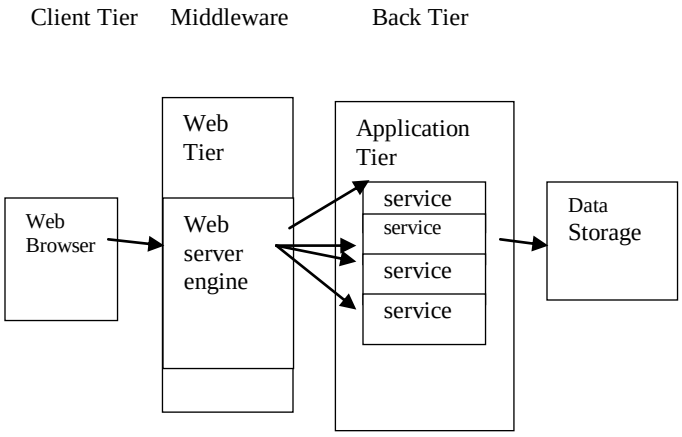

Fig 3: **General Web Application**

The proposed solution to implement the Web Service Engine (WSE) as introduced above covers several individual components and a service management layer which can easily be adapted and plugged into the existing service management of the specific mobile agent platform:

Stub Generator SOAP messages transported through the network as result of a web service invocation are typically exchanged between a client and a server stub. With respect to the WSE architecture, the server stub processes incoming messages and triggers the associated service object by means of direct method invocation. Vice versa, the client implements the service's interface and starts communication with the associated server stub, upon local method invocation. Thereby, the task of the stub generator is twofold. On the server side, it extracts the specific Java interface from a given service object, automatically generates a corresponding syntactic WSDL description and creates a new server stub, which is then associated with the service object. On the client side, it transforms a given WSDL description into the corresponding Java interface, and creates a client stub implementing this interface. To realize automated and transparent integration for Java-based systems, the stub generator must be able to dynamically generate new stub objects during runtime

Web Service Gateway Server stubs created by the stub generator have to be exposed by means of web service endpoints accessible over the network. The web service gateway thereby implements the specific transport protocols and serves as both, web server enabling access to server stubs (e.g. over HTTP and HTTPS) as well as web client used by client stubs as transport layer for the transmission of SOAP messages.

Registry Service To make agent services visible by means of web service discovery, the registry service transforms WSDL descriptions created by the stub generator from a given service object into appropriate UDDI business entities. These business entities are subsequently be registered at a UDDI-compliant registry. Furthermore, this service can be used to search for a web service which is syntactically compatible to a given Java interface.

WSE Service The WSE Service wraps the above described functionality and provides it via a simple interface which can explicitly be used by mobile agents to either search for web services, or to deploy and undeploy encapsulated service objects. In both cases, the agent does not need to know anything about the

traditional web service stack: deployment is done by providing a Java object implementing an arbitrary
Java interface which is automatically exposed by means of a web service, then a search request with a given Java interface directly returns the reference to a client stub implementing this interface, if successful.

Service Management Layer The service management layer transparently activates the above described processes by automatically forwarding appropriate requests (to
register or lookup an agent service within the agent infrastructure) to the WSE service. Since web service deployment and undeployment is subsequently done implicitly, the administrator of the local agent server can configure this layer in advance, and select the types of services to automatically expose as web services.

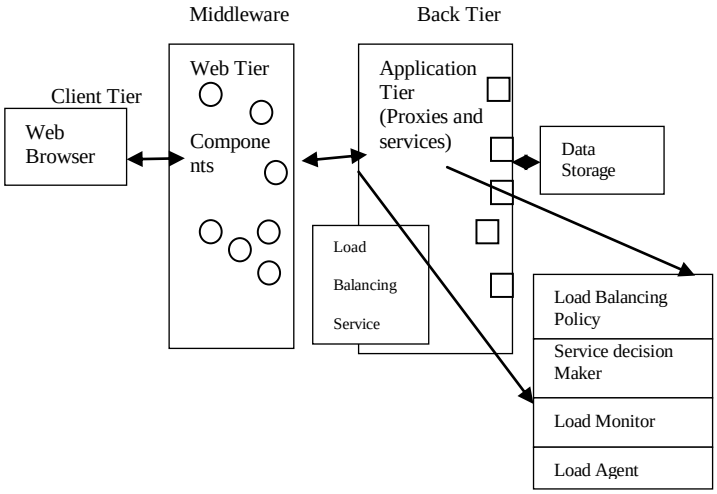

Fig 4: Load Balancing in Web Application

Interface to WAP

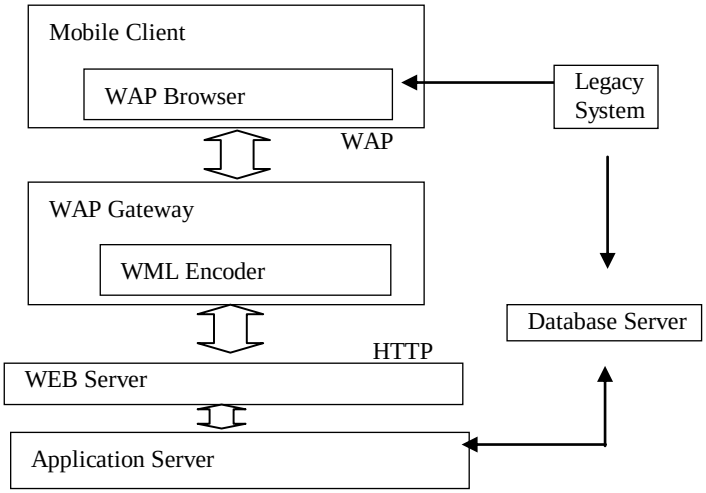

Fig 5: **Distributed Architecture of WAP based Mobile**

WAP 2.0 brings the wireless world closer to the Internet with a suite of specifications that utilizes technologies that will enhance the wireless user experience.
With the release of WAP 2.0, the WAP Forum has successfully accomplished several objectives:

1. Add support for the standard Internet communication protocols. WAP 2.0 provides support for protocols such as IP, TCP and HTTP. By adding these Internet protocols and standards and providing interoperable optimizations suitable to the wireless telecommunications environment, the WAP specifications provide an environment that permits wireless devices to utilize existing Internet technologies.
2. Provide a rich application environment, which enables delivery of information and interactive services to digital mobile phones, pagers, personal digital assistants (PDAs) and other wireless devices.
3. Minimize the use of device processing power and optimize network resources in order to minimize costs and maximize performance.

The following items represent the major architectural components of WAP 2.0:

Protocol Stack Support – In addition to the WAP Stack introduced in WAP 1, WAP 2.0 adds support and services
on a stack based on the common Internet stack including support for TCP, TLS and HTTP. By encompassing both
stacks, WAP 2.0 provides a connectivity model on a broader range of networks and wireless bearers.

WAP Application Environment – Nominally viewed as the 'WAP Browser', the WAP 2.0 Application Environment has evolved to embrace developing standards for Internet browser markup language. This has led to the definition of the XHTML Mobile Profile (XHTMLMP). XHTMLMP is based on the modularity framework of the eXtensible HyperText Markup Language (XHTML) developed by the W3C to replace and enhance the currently used HTML language common today. The use of Internet technologies is not new for WML, as WML1 is a fully conformant XML language in its own right.

Additional Services and Capabilities – The WAP specifications have had items that were neither part of the
'WAP Stack' nor the 'WAP Browser' but helped to enrich the environment defined in the WAP specifications. With
WAP 2.0, there is a considerable increase in the number of features available to developers, operators and users.

WAP 2.0 capitalizes on a wide range of new technologies and advanced capabilities, such as:

Networks and Network Bearers – Carriers worldwide are upgrading their existing networks with higher-speed bearers such as General Packet Radio Service (GPRS) and High-speed Circuit-Switched Data (HSCSD) and introducing higher bandwidths and speeds in third-generation (3G) wireless networks such as W-CDMA and CDMA2000 3XRTT. These higher capable network bearers permit new types of content (e.g., streaming media) and

provide an 'always on' availability. These new aspects of the serving networks permit new operational activities.

TCP/IP as Transport Protocol – Most new wireless network technologies provide IP packet support as a basic data transport protocol. WAP 2.0 leverages IETF work in the Performance Implications of Link Characteristics (PILC) Working Group to develop a mobile profile of TCP for wireless links. This profile is fully interoperable with the 'common' TCP that operates over the Internet today.

Processors – Manufacturers continue to introduce smaller devices with faster and more power-efficient processors and dipoles that are higher-definition and in color. Additionally, more efficient packaging technology permits smaller integrated circuits and more sophistication in a given size of device. The net effect is that new wireless devices have more capabilities that can be leveraged to enhance the services delivered to the user.

Mobile-friendly Technologies – With the growth in usage of mobile devices, there is an increased awareness of the needs specific to the mobile user. The WAP Forum has worked with the W3C and the IETF to help characterize the key issues that impact wireless usage of the web. Through that involvement, and from the interest of their own membership,
The W3C has lately presided over advances in more mobile-friendly technologies, including:

1. The release in late 2000 of the recommendation for the Basic profile for the Extensible Hypertext Markup language (XHTML). This Basic profile incorporates the core elements of the XHTML language, which provides a framework for expandability and enhancement.
2. Recent updates to the Composite capabilities/Preference profiles (CC/PP) framework for describing user preferences and device capabilities. CC/PP provides the technical basis for the UAPROF device profile function.
3. The release of the Cascading Style Sheets (CSS) Mobile Profile provides a subset of CSS version 2 that is targeted at devices such as smart phones, personal digital assistants (PDAs) etc.

M. CONCLUSION

Algorithm stability, which is precondition to scalability, is an indication of the ability of the algorithm to avoid poor allocation decisions. To assess stability we can measure hit-ratio, the ratio of remote execution requests concluded successfully. Another measure of stability is percentage of remote execution in the system. Activities related to remote execution should be bounded and restricted to a small proportion of the activity in the system.

N. REFERENCES

[1] K.Q. Yan1, S.C. Wang1 "The Anatomy Study of Load Balancing in Distributed System**"**, proceeding of the seventh international conference on parallel and distributed computing, Application and Technologies (PDCAT'06)

[2] Reinhard Riedl and Lutz Richter "Classification of Load Distribution Algorithms ", CH-8057,1066-6192/96 $5.00 *0* 1996 IEEE Proceedings of PDP '96

[3] Ka-Po Chow and Yu-Kwong Kwok "On Load Balancing for Distributed Multiagent Computing", ieee transactions on parallel and distributed systems, vol. 13, no. 8, august 2002

[4] Rami G. Melhem,Kirk R. Pruhs, and Taieb F. Znati"Using Spanning-Trees for Balancing Dynamic Load on Multiprocessors", ieee transactions on parallel and distributed systems, vol. 06, no. 8, march 1996

[5] Rome Tor Vergata ,PHILIP S. YU IBM T.J. "Dynamic Load balancing on web server system", ieee internet computing, no. 8, may-june 1999

[6] Cho Cho Myint, Khin Mar Lar Tun "A Framework of Using Mobile Agent to Achieve Efficient Load Balancing in Cluster "University of Computer Studies Yangon, Myanmar

[7] K. Yang, X. Guo, A. Galis, B. Yang, and D. Liu, "Towards Efficient Resource On-Demand in Grid Computing," ACM SIGOPS Operating Systems Review, Vol. 37, No. 2, pp. 37-43, 2003

[8] O. Kremien, J. Kramer " Methodical Analysis of Adaptive Load Sharing Algorithms" IEEE Transactions on Parallel and Distributed Systems, Vol 3, No.6 November 1992.

[9] Yiqiang Zheng, Heqing Guo, Wei Gao, Botong Xu "Research on Load Balance of Multi Clusters Architecture Based on Business Components Partition "Proceedings of the Third International Conference on Information Technology and Applications (ICITA'05) 0-7695-2316-1/05 © 2005 IEEE

[10] Qi Zhang ,Ningfang Mi, Alma Riska " Load Unbalancing to Improve Performance under Autocorrelated Traffic" Proceedings of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS'06) 0-7695-2540-7/06 © 2006 IEEE

[11] Thomas L Consvent ,John G Kuhl " **A** Taxonomy of Scheduling in General-Purpose Distributed Computing Systems" ieee transactions on software engineering, vol. **14,** no. 2, february 1988

[12] Orly Kremien and Jeff Kramer " Methodical Analysis of Adaptive Load Sharing Algorithms "IEEE TRANSACTIONS ON Parallel and distributed systems, VOL. 03**,**NO. 6, November 1992

[13]] Bhaskaran Raman, Randy H. Katz, "Load Balancing and Stability Issues in Algorithms for Service Composition" ieee infocom 2003

[14] Chi-Chung Cheung Man-Ching Yuen Angus C H Yip "Dynamic DNS for Load Balancing "Proceedings of the 23 rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03) 0-7695-1921-0/03 © 2003 IEEE

[15] Robert Bialek Eric Jul "A Framework for Evolutionary, Dynamically Updatable,Component-based Systems " Proceedings of the 24th International Conference on Distributed Computing Systems Workshops (ICDCSW'04) 0-7695-2087-1/04 © 2004 IEEE

[16] Dariusz Kowalski Peter M. Musiał Alexander A. Shvartsman "Explicit Combinatorial Structures for Cooperative Distributed Algorithms "Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICSCS'05) 1063-6927/05 .00 © 2005 IEEE

[17] Hugo Miranda ,Lu´ıs Rodrigues "Using a Fairness Monitoring Service to Improve Load-Balancing in DSR"

Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05) 1545-0678/05 © 2005 IEEE

[18] Thomas Koch, Gerald Rohde, Bernd Kramer" Adaptive Load Balancing in a Distributed Environment" 0-8186-5835-5/94 $03.00 63 1994 IEEE

[19] Valeria Cardellini,Michele Colajanni,Philip S. Yu "Redirection Algorithms for Load Sharing in Distributed Web-server Systems" Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICSCS'05) 1063 6927/05 .00 © 2005 IEEE

[20] Xiaodong Zhang1 Yanxia Qu2 Li Xiao "Improving Distributed Workload Performance by Sharing Both CPU and Memory Resources"

[21] Xiaodong Lu, Yi Zhou and Kinji Mori "Agent-Based Rating Oriented Information Provision and Reallocation for High-Assurance in Open and Dynamic Environments " Proceedings of the 24th International Conference on Distributed Computing Systems Workshops (ICDCSW'04) 0-7695-2087-1/04 $20.00 © 2004 IEEE

[22] Adnan Agbaria William H. Sanders "Application-Driven Coordination-Free Distributed Checkpointing*"* Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICSCS'05) 1063-6927/05 © 2005 IEEE

[23] Bhaskaran Raman, Randy H. Katz "Load Balancing and Stability Issues in Algorithms for Service Composition "0-7803-7753-2/03/$17.00 (C) 2003 IEEE

[24] Erik Putrycz nad Guy Benard "Client Side Reconfiguration on software components for Load Balancing" 0-7695-1080-9/01 $10.00 (C) 2003 IEEE

[25] Asma Ben Letaifa, Sami Tabban, Zied Chou" A Hybrid Algorithm to Reconfigure platforms of Radio Mobile Services" Proceedings of the IEEE Conference on Local Computer Network 30th Anniverssary (LCN'05) 1063 - 927/05 .00 © 2005 IEEE

[26] Pradeep K Sinha " Distributed Operating systems : concepts and Design "IEEE computer society Press, Prentice Hall India - 2004

[27] Mukesh Singhal, Niranjan G. Shivaratri "Advanced Concepts in Operating Systems"

[28] George Coulouris, Jean Dollimore, Tim Kindeberge "Distributed Systems: Concepts and Design", Pearson Education

[29] Bhardwaj, Ghosh , Mani" Scheduling Divisible loads in parallel and distributed system"

[30] Shiraji, Hurson Kav. " Scheduling and Load Balancing in parallel and distributed System"

[31] "The Next Step In Server Load Balancing" Alteon ebSystems, Inc. 50 Great Oaks Boulevard San Jose, California 95119 408-360-5500408-360-5501 http://www.alteon.com

[32] "Web Service Scalability and Performance with Optimising Intermediaries "*Mark Nottingham*

[33] Ananya Das, Charles Martel, Biswanath Mukherjee, and Smita Rai "A Better Approach to Reliable Multi-Path Provisioning" Department of Computer Science, University of California, Davis, CA 95616 Email: *f*das, martel, mukherje, rai*g*@cs.ucdavis.edu

[34] "Works in Progress: The 2nd International Middleware Doctoral Symposium" IEEE DISTRIBUTED SYSTEMS ONLINE 1541-4922 © 2006 Published by the IEEE Computer Society Vol. 7, No. 3; March 2006

[35] VALERIA CARDELLINI Tor Vergata PHILIP S. YU IBM T.J "DYNAMIC LOAD BALANCING ON WEB-SERVER SYSTEMS ".

[36] Wang Fangxiong, Jiang Zhiyong "Research on A Distributed Architecture of Mobile GIS Based on WAP",State Key Laboratory of Information engineering in Surveying, Wuhan University

[37] "WAP Forum : Wireless Application Protocol, Technical White Paper" January 2002, www.wapforum.org