Load Balancing in Mobile Ad Hoc Networks by Using Different Routing Protocols and Algorithms

Minakshi
Department of Computer Science & Engineering
Sai Institute of Engineering and Technology
Amritsar, 143001

Tanu Preet Singh
Research Scholar Student,
Uttarakhand Technical University,
Dehradun, India

Prof. R.K Singh
Professor & OSD
Uttarakhand Technical University
Dehradun, India

Abstract: An ad-hoc network consists of a set of mobile nodes which are connected with each other by using radio waves. Load balancing is the process of improving the performance of a parallel. This network does not have any infrastructure or central administration, hence it is called infrastructure less network. As the nodes are mobile, it is very difficult to find the path between two end points. This paper presents a solution for finding path between nodes in mobile ad hoc network. For maintaining multiple routes between two endpoints on top of the Stream Control Transmission Protocol (SCTP), and the Dynamic Source Routing (DSR) protocol. A number of additional modifications are incorporated to the SCTP protocol in order to allow its smooth operation. Some of the parameters used to evaluate its performance are packet delays and throughput. The results of this algorithm shows better throughput as compared to existing algorithms. In this paper we present the performance analysis of various load balancing algorithms based on different parameters, considering two typical load balancing approaches static and dynamic. The analysis indicates that static and dynamic both types of algorithm can have advancements as well as weaknesses over each other. Deciding type of algorithm to be implemented will be based on type of parallel applications to solve. The main purpose of this paper is to help in design of new algorithms in future by studying the behavior of various existing algorithms.

Keywords—Load balancing (LB), SCTP, DSR, distributed systems, Static Load balancing, Dynamic Load Balancing

1. INTRODUCTION

Mobile Ad hoc network is self configuring network of mobile hosts connected by wireless links, the union of which forms the topology of the network [1]. The advantages of ad hoc networks are the convenience (no central administration), mobility, productivity, deployment and expandability. As the nodes in the network are mobile, the topology of network changes unpredictably. Hence it is difficult to generate path between two nodes. This paper deals with the development of on-demand ad-hoc network routing which can achieve load balancing for packet switched network. The algorithm is adaptive, distributed and is inspired by swarm intelligence. Ant algorithms are the class of optimizing algorithms under swarm intelligence (SI)[2][3]. Routing in ant algorithm [4][5] is through interaction of network exploration agents called ants. According to this algorithm, a group of mobile agents builds path between pairs of nodes by exchanging information and updating routing tables. MANET networks have several usages. First these networks were devised to be used in military applications. MANET networks are mostly used in survey, helping and saving operations, tracing and operations, scientific conferences.

Proc. of the Intl. Conf. on Advances in Electronics, Electrical and Computer Science Engineering — EEC 2012
Edited by Dr. R. K. Singh.

ISBN: 978-981-07-2950-9

The problem of mobile ad-hoc network (MANET) can be summarized in the answer of this question: how to find the route between the communicating end-points. One of the main reasons is that routing in MANETs is a particularly challenging task due to the fact that the topology of the network changes constantly and paths which were initially efficient can quickly become inefficient or even infeasible. Moreover, control information in the network is very restricted. This is because the bandwidth of the wireless medium is very limited, and the medium is shared. It is therefore important to design algorithms that are adaptive, robust and self-healing. Moreover, they should work in a localized way, due to the lack of central control or infrastructure in the network [6,8].
A major challenge this work faces is to provide an appropriate localization-free1 definition of the center of the network, using the topology information available at every node. Since the topology information may be exhaustive (proactive protocols) or partial (reactive protocols), we had to consider each case separately. The main goal is to distribute the jobs among processors to maximize throughput, maintain stability, resource utilization and should be fault tolerant in nature. Local scheduling performed by the operating system consists of the distribution of processes to the time-slices of the processor. On the other hand Global scheduling is the process of deciding where to execute a process in a multiprocessor system.

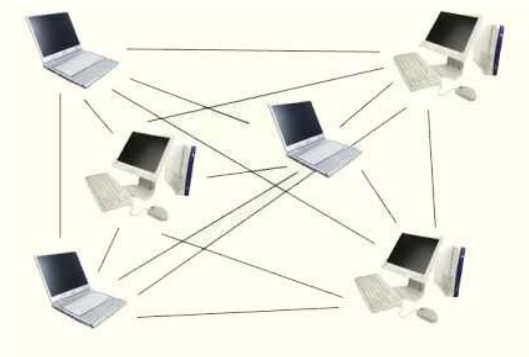

Figure1: MANET

2. OVERVIEW OF PROTOCOLS

A simple DSR is a simple source routing protocol for MANETs, in which route caching is heavily used. If the route to the destination is not known, a route discovery process is initiated in order find a valid route. Route discovery is based in flooding the network with route request (RREQ) packets. Every mobile host that receives a RREQ packet checks the contents of its route cache, and if it is the destination or it has a route to the destination it replies to the RREQ with a route reply (RREP) packet that is routed back to the original source. In case none of the above holds, the host that received the RREQ re-broadcasts it to its neighborhood. In this way the RREQ message is propagated till the destination. Note that both RREQ and RREP packets are also source routed. The RREQ message maintains the path traversed across the network allowing thus the RREP message to route itself back to the source by traversing the recorded path backwards. The route carried back by the RREP packet is cached at the source for future use. If any link on a source route is broken, the source host is notified with a special route error (RERR) packet from intermediate nodes. When the source gets this packet removes any route using this link from its cache. More details and enhancement to this basic DSR operation can be found in [9].

SCTP was recently adopted by IETF, and is a reliable transport protocol that operates on top of a connectionless packet based network such as IP. One of the most important new ideas that SCTP introduced is that of multi-homing. A single SCTP association (session), is able to use alternatively anyone of the available IP-addresses without disrupting an ongoing session. However, this feature is currently used by SCTP only as a backup mechanism that helps recovering from link failures.
SCTP maintains the status of each remote IP address by sending Heartbeat messages and it is thus able to detect a specific link failure and switch to another IP address. Another novel feature is that SCTP decouples reliable delivery from message ordering by introducing the idea of streams. The stream is an abstraction that allows applications to preserve in order delivery within a stream but unordered delivery across streams. This feature avoids HOL blocking at the receiver in case multiple independent data streams exist in the same SCTP session. Congestion control was defined similar to TCP, primarily for achieving TCP friendliness [10].
In this paper, we propose two methods to Improve the Ad-Hoc On-Demand Distance-Vector (**AODV**) protocol. The main goal in the design of the protocol was to reduce the routing overhead, buffer overflow, end-to-end delay and increase the performance. A multi-path routing protocol is proposed which is based on AODV

and Ant Colony Optimization(ACO). This protocol is refereed to Multi-Route AODV Ant routing (MRAA). Also we propose a load balancing method that uses all discovered paths simultaneously for transmitting data. In this method, data packets are balanced over discovered paths and energy consumption is distributed across many nodes through network.

3. ALGORITHM

3.1 STATIC LOAD BALANCING

In this method the performance [11] [12] of the processors is determined at the beginning of execution. Then depending upon their performance the work load is distributed in the start by the master processor. The slave processors calculate their allocated work and submit their result to the master. A task is always executed on the processor to which it is assigned that is static load balancing methods are non-preemptive. The goal of static load balancing method is to reduce the overall execution time of a concurrent program while minimizing the communication delays. A general disadvantage of all static schemes is that the final selection of a host for process allocation is made when the process is created and cannot be changed during process execution to make changes in the system load.

A. Round Robin and Randomized Algorithms

In the round robin [13] processes are divided evenly between all processors. Each new process is assigned to new processor in round robin order. The process allocation order is maintained on each processor locally independent of allocations from remote processors. With equal workload round robin algorithm is expected to work well. Round Robin and Randomized schemes [12] work well with number of processes larger than number of processors. Advantage of Round Robin algorithm is that it does not require inter-process communication. Round Robin and Randomized algorithm both can attain the best performance among all load balancing algorithms for particular special purpose applications. In general Round Robin and Randomized are not expected to achieve good performance in general case.

B. Central Manager Algorithm

In this algorithm [14], A central processor selects the host for new process. The minimally loaded processor depending on the overall load is selected when process is created. Load manager selects hosts for new processes so that the processor load confirms to same level as much as possible. On hand information on the system load state central load manager makes the load balancing judgment. This information is updated by remote processors, which send a message each time the load on them changes. This information can depend
on waiting of parent's process of completion of its children's process, end of parallel execution. The load manager makes load balancing decisions based on the system load information, allowing the best decision when of the process created. High degree of inter-process communication could make the bottleneck state. This algorithm is expected to perform better than the parallel applications, especially when dynamic activities are created by different hosts.

C. Threshold Algorithm

According to this algorithm, the processes are assigned immediately upon creation to hosts. Hosts for new processes are selected locally without sending remote messages. Each processor keeps a private copy of the system's load. The load of a processor can characterize by one of the three levels: under loaded, medium and overloaded. Two threshold parameters *tunder* and *tupper* can be used to describe these levels.

Under loaded - load < tunder
Medium - tunder ≤ load ≤ tupper
Overloaded - load > tupper

Initially, all the processors are considered to be under loaded. When the load state of a processor exceeds a load level limit, then it sends messages regarding the new load state to all remote processors, regularly updating them as to the actual load state of the entire system. If the local state is not overloaded then the process is allocated locally. Otherwise, a remote under loaded processor is selected, and if no such host exists, the process is also allocated locally. Thresholds algorithm have low inter process communication and a large number of local process allocations. The later decreases the overhead of remote process allocations and the overhead of remote memory accesses, which

leads to improvement in performance. A disadvantage of the algorithm is that all processes are allocated locally when all remote processors are overloaded. A load on one overloaded processor can be much higher than on other overloaded processors, causing significant disturbance in load balancing, and increasing the execution time of an application.

3.2 DYNAMIC LOAD BALANCING

It differs from static algorithms in that the work load is distributed among the processors at runtime. The master assigns new processes to the slaves based on the new information collected [15]. Unlike static algorithms, dynamic algorithms allocate processes dynamically when one of the processors becomes under loaded. Instead, they are buffered in the queue on the main host and allocated dynamically upon requests from remote hosts.

A. Central Queue Algorithm

Central Queue Algorithm [16] works on the principle of dynamic distribution. It stores new activities and unfulfilled requests as a cyclic FIFO queue on the main host. Each new activity arriving at the queue manager is inserted into the queue. Then, whenever a request for an activity is received by the queue manager, it removes the first activity from the queue and sends it to the requester. If there are no ready activities in the queue, the request is buffered, until a new activity is available. If a new activity arrives at the queue manager while there are unanswered requests in the queue, the first such request is removed from the queue and the new activity is assigned to it. When a processor load falls under the threshold, the local load manager sends a request for a new activity to the central load manager. The central load manager answers the request immediately if a ready activity is found in the *process-request queue*, or queues the request until a new activity arrives.

B. Local Queue Algorithm

Main feature of this algorithm [16] is dynamic process migration support. The basic idea of the local queue algorithm is static allocation of all new processes with process migration initiated by a host when its load falls under threshold limit, is a user-defined parameter of the algorithm. The parameter defines the minimal number of ready processes the load manager attempts to provide on each processor. Initially, new processes created on the *main* host are allocated on all under loaded hosts. The number of parallel activities created by the first Parallel construct on the main host is usually sufficient for allocation on all remote hosts. From then on, all the processes created on the main host and all other hosts are allocated locally. When the host gets under loaded, the local load manager attempts to get several processes from remote hosts. It randomly sends requests with the number of local ready processes to remote load managers. When a load manager receives such a request, it compares the local number of ready processes with the received number. If the former is greater than the latter, then some of the running processes are transferred to the requester and an affirmative confirmation with the number of processes transferred is returned.

TABLE I
PARAMETRIC COMPARISON OF ROUND ROBIN AND RANDOM LOAD BALANCING ALGORITHMS

Parameters	**Round Robin**	**Random**
Overload Rejection	No	No
Fault Tolerant	No	No
Forecasting Accuracy	More	More
Process Migration	No	No
Cooperative	No	No
Stability	Large	Large
Resource Utilization	Less	Less

TABLE II

PARAMETRIC COMPARISON OF LOCAL QUEUE AND CENTRAL QUEUE LOAD BALANCING ALGORITHMS

Parameters	**Local Queue**	**Central Queue**
Overload Rejection	Yes	Yes
Fault Tolerant	Yes	Yes
Forecasting Accuracy	Less	Less
Process Migration	Yes	No
Cooperative	Yes	Yes
Stability	Large	Small
Resource Utilization	More	Less

TABLE III
PARAMETRIC COMPARISON OF CENTRAL MANAGAR AND THRESHOLD LOAD BALANCING ALGORITHMS

Parameters	Central Manager	Threshold
Overload Rejection	No	No
Fault Tolerant	Yes	No
Forecasting Accuracy	More	More
Process Migration	No	No
Cooperative	Yes	Yes
Stability	Large	Large
Resource Utilization	Less	Less

4. CONCLUSION

Load balancing algorithms and protocols work on the principle that in which situation workload is assigned, during compile time or at runtime. The above comparison shows that static load balancing algorithms are more stable in compare to dynamic and it is also ease to predict the behavior of static, but at a same time dynamic distributed algorithms are always considered better than static algorithms.

REFERENCES

[1] Andrew S Tannenbaum, "Computer Networks", 4th Edition, Prentice-Hall of India

[2] E. Bonabeau, M. Dorigo, and G. Théraulaz, Swarm intelligence: from natural to artificial systems, Oxford University Press, 1999.

[3] T. White, "Swarm intelligence and problem solving in telecommunications", Canadian Artificial Intelligence Magazine, spring, 1997. International Journal of Next-Generation Networks (IJNGN),Vol.1, No.1, December 2009.

[4] G. Di Caro and M. Dorigo, "Mobile agents for adaptive routing", Proc. 31st Hawaii International Conference on System Sciences, IEEE Computer Society Press, Los Alamitos, CA, pp. 74-83, 1998.

[5] Schoonderwoerd R, Holland O, Bruten J, Rothkrantz L. "Ant-Based load Balancing in telecommunications networks, Adaptive Behavior Hewlelt-Packard Laboratories, Bristol-England, pp 162-207, 1996.

[6] G. Di Caro and M. Dorigo, *AntNet: distributed stigmergetic control for communications networks*, Journal of Artificial Intelligence Research, 9 (1998), 317–365.

[7] G. Di Caro, F. Ducatelle, and L. M. Gambardella, *AntHocNet: an adaptive nature-inspired algorithm for routing in mobile ad hoc networks*, Tech. Report IDSIA-27-04-2004, Dalle Molle Institute for Artificial Intelligence (IDSIA), Manno-Lugano, Switzerland, September 2004.

[8] M. Dorigo and G. Di Caro, *The ant colony optimization metaheuristic*,in New Ideas in Optimization, D. Corne, M. Dorigo, and F. Glover, eds., McGraw-Hill, London, UK, 1999, 11–32. [8] Macker. J and Corson. S, Mobile ad hoc networks (MANET), 1997, http://www.ietf.org/html.charters/manet/charter.html.

[9] D.B. Johnson, D.A. Maltz, Y.-C. Hu, The dynamic source routing protocol for mobile ad hoc networks (DSR). Available from: <http://www.ietf.org/internet-drafts/draftietf-manet-dsr-09.txt>. [10] Toh.C-K., Ad Hoc Mobile Wireless Networks: Protocols and Systems (Prentice-Hall, New York, 2002).

[10] R.R. Stewart et al., Stream control transmission protocol, RFC 2960, October 2000.

[11] Derek L. Eager, Edward D. Lazowska , John Zahorjan, "Adaptive load sharing in homogeneous distributed systems", IEEE Transactions on Software Engineering, v.12 n.5, p.662-675, May 1986.

[12] R. Motwani and P. Raghavan, "Randomized algorithms", ACM.

[13] Zhong Xu, Rong Huang, "Performance Study of Load Balancing Algorithms in Distributed Web Server Systems", CS213 Parallel and Distributed Processing Project Report.

[14] P. L. McEntire, J. G. O'Reilly, and R. E. Larson, Distributed Computing: Concepts and Implementations. New York: IEEE Press, 1984.

[15] S. Malik, "Dynamic Load Balancing in a Network of Workstation", 95.515 Research Report, 19 November, 2000.

[16] William Leinberger, George Karypis, Vipin Kumar, "Load Balancing Across Near-Homogeneous Multi-Resource Servers", 0-7695-0556-2/00, 2000 IEEE.