ISSN: 2321-7782 (Online)
Volume 3, Issue 6, June 2015
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

An Efficient Method for Improving Quality of Service using TCP/IP Incast Congestion Model

Praveena. N[1]
Asst. Professor,
Department of IT,
Alpha College of Engg. & Tech,
Pondicherry - India

Samundeeswari. P[2]
Asst. Professor,
Department of ECE,
Alpha College of Engg. & Tech,
Pondicherry - India

Saraladevi. D[3]
Department of IT,
Alpha College of Engg. & Tech,
Pondicherry - India

Abstract: ***Transport Control Protocol (TCP) incast congestion happens when number of senders work in parallel with the same receiver where the high bandwidth and low latency network problem occurs. For many data center network application such as many to one, heavy traffic is present on such server. Incast congestion degrades the performance as the packets are lost at server side due to buffer overflow and hence the response time will be more. To improve the performance, main idea is to design an incast congestion Control for TCP (ICTCP) scheme on the receiver side by implementing grid computing for process allocation as well as sub server scheduling to achieve flow shop scheduling process (FSSP). Mobile agent is used to handle failures thus the specific part of failed file will be transmitted rather than whole part to be sent. Hashing technique is used for message integrity and in order to improve the quality of service fault tolerance system is used. To avoid the packet loss the ICTCP algorithm has been implemented.***

Keywords: ***Data-center networks, Incast congestion, FSSP, ICTCP, Grid computing, Mobile Agent.***

I. INTRODUCTION

Multiple synchronized servers send data to a same receiver in parallel, TCP incast congestion happens in high bandwidth and low latency networks. Many to one traffic pattern is common in Map reduce and Search data center applications. In this paper, our discussion is mainly on the congestion problem that usually occurs in the incast. Incast is condition that is opposite to that of that of the broadcast. In broadcasting one node sends out messages to the multiple nodes but in Incast, multiple nodes send messages to the same node. Network congestion is the situation in which an increase in data transmissions results in a proportionately smaller increase [1], or even a reduction, in throughput. Congestion degrades the performance of the network. Packets loss is the major reason for Congestion. The root cause of TCP incast collapse is that the highly burst traffic of multiple TCP connections overflows the Ethernet switch buffer in a short period of time which produces intense packet loss, thus causes TCP timeouts and retransmission. Focus is on avoiding packet loss before incast congestion which leads more attractive than recovery after loss. Our idea is to perform incast congestion avoidance at the receiver side by preventing incast congestion. The natural choice is receiver side since it knows the throughput of all TCP connections and the available bandwidth. The receiver window size can be adjusted at the receive side of each TCP connection so that aggregate burstiness of all the synchronized senders are kept under control. Our aim is to design incast congestion Control for TCP (ICTCP)[2]. However, controlling the receive window is difficult, the receive window should be kept small enough to avoid incast congestion, but also large enough for good performance and other non incast cases.

The technical novelties of this work are as follows: 1) In distributed file systems, the files are deliberately stored in multiple servers. Still TCP incast congestion occurs when multiple blocks of a file are fetched from multiple servers at the same time. To

perform congestion control on the receiver side, we use grid computing for process allocation as well as sub server scheduling.2) the per-flow congestion control is performed independent of slotted time of the round-trip time (RTT) of each connection has control latency in its feedback loop. 3) Mobile agent is used to handle failures thus the specific part of failed file will be transmitted rather than whole part to be sent. 4) Hashing technique is used for message integrity [3] and in order to improve the quality of service fault tolerance system is used. To avoid the packet loss, ICTCP algorithm has been implemented.

II. PROPOSED ALGORITHM

In proposed system, the main idea is to avoid incast congestion during data transfer which results in performance degradation in the network. To avoid incast congestion, grid computing is implemented for sub server allocation. To minimize the idle time and waiting time, FSSP algorithm is implemented. To ensure message integrity between the sender and the receiver, CBU hashing algorithm is implemented.

A. Flow Shop Scheduling Process Algorithm

The main use of Flow Shop Scheduling Process (FSSP) algorithm is to maintain a continuous flow of processing tasks which is desired with a minimum of idle time and a minimum of waiting time. FSS is a special case of job shop scheduling where there is strict order for all operations to be performed on all jobs. The working of algorithm is to split the finite set of "n" jobs, which consists of chain of operations [4]. A finite set of "m" machines, where each machine can handle at most one operation at a time .Each operation needs to be processed during an uninterrupted period of a given length on a given machine. The main Purpose of FSSP algorithm is to reduce the latency, when there is multiple sender and receivers in parallel.

The problem may have some constraints at the start and delivery time for each job. The objective of solving this problem is to identify the flow sequences on machines in order to optimize the performance criteria. One of the following objectives may occur in job shop scheduling problem:

1. Maximize the utilization of systems or resources, in other words to minimize the latency and congestion.
2. Minimize the operational or the idle time in each machine, caused by the earliness of operation process or tardiness in receiving the jobs by a machine.

B. CBU Hashing Algorithm

In proposed system, hashing algorithm is implemented for ensuring message integrity between the sender and the receiver. The goal in choosing any hashing algorithm is to spread out the records as uniformly as possible over the range of addresses available in the storage space. In this system, Cost Based Unification (CBU) hashing technique [5] is used. It acts as a better algorithm since it provides 12 bits of security against collision attacks. The working of algorithm is, initially converts the original value into byte code and the byte code is verified. After verification of code, the corresponding value is generated as the byte code. The value which is generated for the byte code is attached to the value along with the message given from sender. On the receiver side, again the above 3 process will be repeated. Finally the integrity is achieved by matching the generated value from the sender and receiver.

III. SOLVING FSSP USING GRID COMPUTER

At its most basic level, grid computing is a computer network in which each computer's resources are shared with every other computer in the system. The memory, processing power and data storage are all community resources that authorized users can tap into and leverage for specific tasks. A grid computing system can be as simple as a collection of similar computers running on the same operating system or as complex as inter-networked systems comprised of every computer platform one can think of.

All the machines or systems that are going to execute the jobs are connected in a LAN network. Out of these systems one system is designated as the Main Server[6,7], and all the other systems are termed as Sub Servers. The request for a Job to be processed comes to the Main Server. It then allots this work to one of the Sub Server for processing.

- *Use of Heuristics in Grid Computing*

The status of all the Sub Server is being maintained locally and by the Main Server in a Database. The status of a Sub Server can include idle, Normalized, Overloaded etc. Based on the status of these the requested job is allocated to one of the Sub Server[8]. The status needs periodical updating to be done by both the Sub and the Main servers.

The use of Grid Computing improves the performance of the JSSP when a situation comes where a job needs to be transferred from one system to another. The reason to do this may be many from a system becoming unavailable, or a system getting overloaded. The method proposed [9] gives a method for shifting of jobs when the resources become unavailable. But the method proposed gives a solution that is restricted to the situation where the availabilities of the resources have to be known in advance and fixed.

- *The Mobile Agent – Aglet*

In computer, a mobile agent is a composition of computer software and data which is able to migrate (move) from one computer to another autonomously and continue its execution on the destination computer. Agents are used in grid and for grid resource management systems.

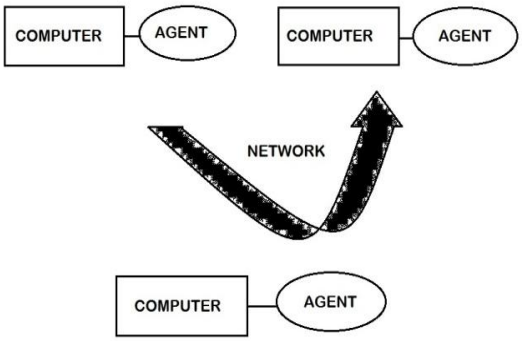

Fig. 1 Mobile Agent transferring data from one computer to another

In fig 1, mobile agent has the unique ability to transport itself from one system in a network to another in the same network. This ability allows it to move to a system containing an object with which it wants to interact and then take advantage of being in the same host or network as the object. A Mobile Agent, namely, is a type of Software Agent.

More specifically, a mobile agent is a process that can transport its state from one environment to another, with its data intact, and be capable of performing appropriately in the new environment. Mobile agents decide when and where to move.

Aglet is a library written in Java, released by IBM to support the development of mobile code. The execution environment within which Aglets are executed is referred to as the Aglet's Context and is responsible for enforcing the security restrictions of the mobile code.

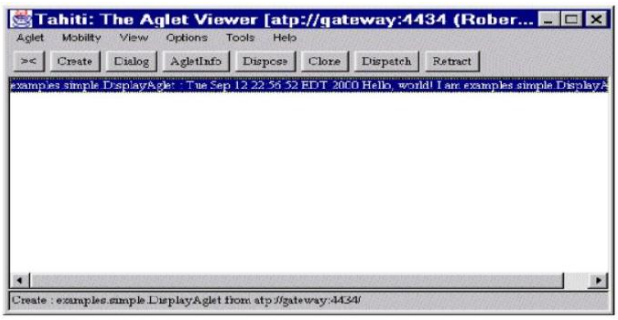

Fig. 2 The Aglet Software is created and execution

- *Tahiti Server*

Aglets run on the Tahiti Server and may adopt seven different states during their life cycle Tahiti is an application program that runs as an agent server. It is possible to run multiple servers (Tahiti) on a single computer by assigning them different port numbers. Tahiti provides a user interface for

- monitoring,
- creating,
- dispatching,
- disposing of agents
- For setting the agent access privileges for the agent server.

The following figure gives an example of a Tahiti Server; the Aglet Software is created and executed on this server.

TABLE 1
Comparison of the proposed method with the existing

EXISTING SYSTEM	PROPOSED SYSTEM
PERFORMANCE	**PERFORMANCE**
1. Sending request from many to one receiver in parallel.	1. Sending request from many to many receivers in parallel.
2. Retransmission of whole file will be done in many to one transmission.	2. Failure part of part is only retransmitted.
3. Subserver status allocation has not been done which results in congestion.	3. Subserver status allocation has been handled which reduces the congestion.
4. Each of the above disadvantages can degrade the system performance.	4. Since they are used in a combination one disadvantage is overcome by the other.
RELIABILITY	**RELIABILITY**
5. Nothing can be done when System gets overloaded.	5. At such a situation, the job can be scheduled by the subservers.
6. Because of this the client cannot have guarantee that a given job would be processed.	6. Here the client can have guarantee that a given job would be processed.
FAULT – TOLERANT	**FAULT – TOLERANT**
7. Nothing can be done when a System failure happens.	7. The Aglet takes care of such a situation.
8. An assumption is made that all resources are continuously available for a job independently of others.	8. No such assumption is made.
9. This assumption will not hold well in Real time situations.	9. Because of no such assumption followed, it can be used in any real time situation.

IV. COMPARISON PROPOSED METHOD WITH EXISTING

Transport Control Protocol (TCP) incast congestion happens when number of senders work in parallel with the same server where the high bandwidth and low latency network problem occurs. Incast congestion degrades the performance as the packets are lost at server side due to buffer overflow and hence the response time will be more. To improve the performance grid computing is implemented at the receiver side for process allocation as well as sub server scheduling to achieve of flow shop scheduling process (FSSP)[10]. Mobile agent is used to handle failures thus the specific part of failed file will be transmitted

rather than whole part to be sent. The proposed method could be applied to any job request and job processing application. Any system involving Grid Computing Architecture is an appropriate place for its implementation. The following table gives a comparison of the proposed method with the existing.

V. SYSTEM ARCHITECTURE

Systems are created to solve problems. One can think of the systems approach as an organized way of dealing with a problem. A collection of components that work together to realize some objectives forms a system. In a system the different components are connected with each other and they are interdependent. This is depicted by the System Architecture diagram. The following figure shows this for solving the FSSP problem.

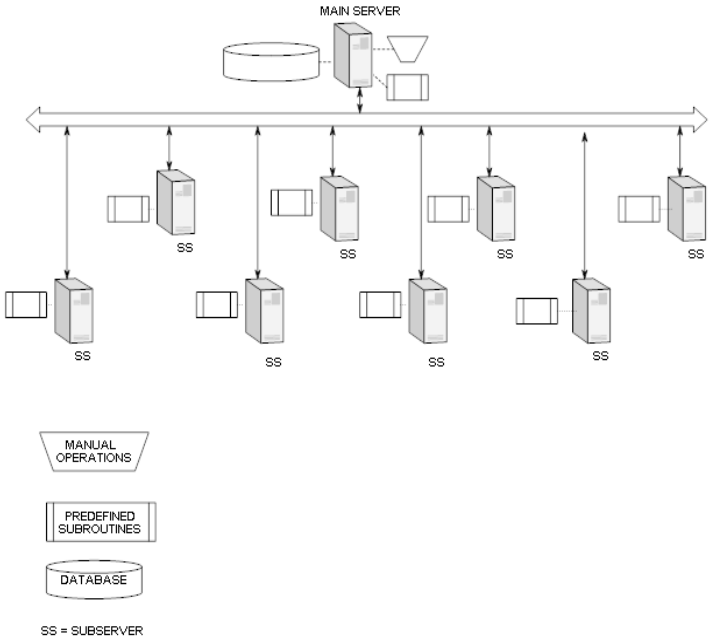

Fig. 3 System Architecture

The Main Server, Sub server comprises the System Architecture. The working of this would be as follows:

The Main Server has the authority to perform these using Mobile Agents:

- Maintains the Database
- Register all Sub Servers details
- Register all Client details
- Store the status information of all sub servers
- Get Request for job processing from clients
- Allocate the Jobs to the sub servers based on their status information.
- It performs Manual operations, has Predefined Sub Routines stored.

The Sub Server performs these:

- These are created by Main server
- After creation these are registered to Main Server
- Performs the job allocated to it by the Main Server
- Stores its own status information in a local Flat file

- When it faces any problem it notifies the Main Server
- It has only the Predefined Sub Routines

VI. DATA FLOW DIAGRAM

In order to determine the working flow of the proposed system, dataflow diagram is depicted. Initially, the Client gives a Request for the job to be processed to the Main Server and heuristics is applied to the Job that is allocated. The Main Server delegates the job to one of the sub server based on its status information where the Sub Server can be in any status such as Idle / Overload / Normal etc. The main work of the Sub server is to process the jobs allocated to it and the main server sends the processed jobs to the Clients as Response[11]. All these functions are performed by using the Mobile Agent.

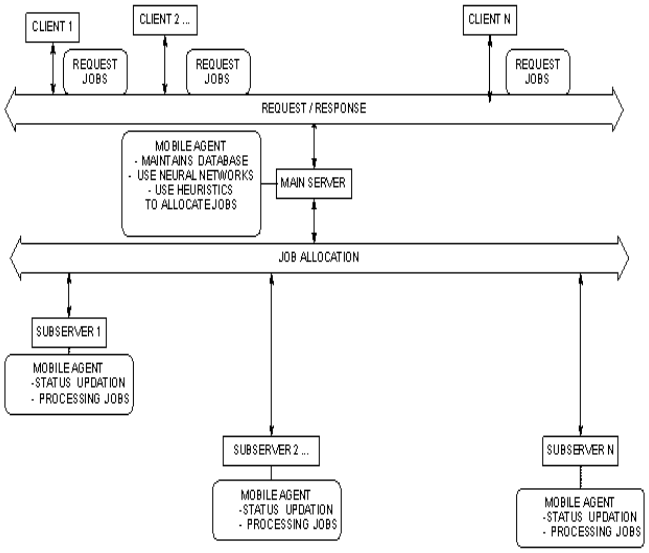

Fig. 4 Data Flow Diagram

VII. IMPLEMENTATION

The different modules that are to be used and their descriptions are follows:

- Client/Server creation
- Sub server creation
- Database creation
- Grid Implementation
- Implementing Hashing algorithm
- Fault Tolerance

1) Server creation – Main server is created such that it will register all the sub server details and the client details. It will allocate jobs to the sub server according to the status checked by it. It will get jobs from the clients and it will allocate the job to a particular sub server.

2) Sub server creation – Sub servers are created and its details are registered to the main server. It will perform the jobs that are allocated to it. It will maintain the status of its own in locally stored flat files. When it faces any problem it will report it to the main server.

3) Database creation – Tables are created in the Oracle database. This database consists of registered clients and sub servers. It will contain the status of all the systems. It will also contain the heuristics once generated.

4) Applying heuristics – The heuristics method is applied to the job allocation at first. It will check all the known conditions for job allocations. The heuristics will be maintained in the database.

A. Use of Flat Files

The above describes the way how the FSSP would be solved. An addition could be made in the place where we make use of a database. The Main server and the Sub server would maintain a database where all the modifications, updating is done. But since this may occur frequently it may degrade the performance of the database which may slow the system performance as a whole. To reduce this we may use a Flat File. Each of the main and sub server can have a flat file for temporary storage and after a particular duration the database can be updated.

VIII. Simulation Study

A Simulation Study is made by generating random numbers using a java code. The time unit used is Minutes. The mean arrival time is generated for the jobs of when they come in to the system. The processing times required by the jobs are given. The sub server that is being allocated to process that job is given. The idle time of the sub server and the idle time of the jobs are calculated.

For the simulation study three sub servers (s1, s2, s3) are taken with the capacities as follows: s1(70), s2(100), s3(40). This meaning that the sub server1 (s1) has a processing capacity of 70 minutes. The Scheduling policy followed is FCFS (First Come First Served).

When jobs come in to the system, the jobs are allocated to the Sub servers based on the processing time needed by the jobs and the capacities of the various sub servers. The following table gives the simulation study done for 6 jobs.

TABLE 2
Results of simulation for 6 jobs

S.No	Mean arrival time of jobs.	Duration needed	Sub server allocated	Processing time	Idle time of sub server	Idle time of sub server
1.	5, J1	30	S3	5 – 35	S3 – 4	
2.	8, J2	60	S1	8 – 68	S1 – 7	
3.	12, J3	40	S2	12 – 52	S2 – 11	
4.	20, J4	20	S3	36 – 56		J4 – 16
5.	25, J5	50	S2	52 – 103		J5 – 28
6.	30, J6	10	S3	57 – 67		J6 – 27

In the table 2, J1, J2 … J6 denotes the jobs that come into the system for processing. And the S1, S2, S3 denotes the three sub servers that are connected to the main server through Grid Computing. The mean arrival time of jobs denotes the time at which the jobs come in to the system for processing.

The average idle time is calculated and given below

- The average idle time of the sub servers

  = 22/ 6 = 3.67 Minutes

- The average waiting time of the jobs

  = 71 / 6 = 11.83 Minutes

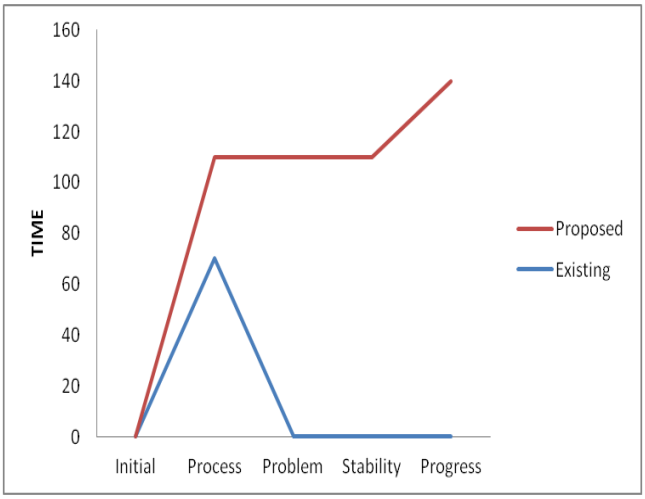

Fig. 5 Comparison of existing and proposed system

Based on the simulation study done it can be concluded that the optimization time or the floating time of the jobs have been decreased. In the literature for 2 jobs and 3 machines the optimization time was 22.2. The following graph shows the merits of the new model being developed.

The above graph shows the main difference between the existing and the proposed system. The proposed system combines both these together. Because of this combination the limitation of one is overcome by the other.

The parameters by which the comparison are done are initial, process, problem, stability and progress. These parameters form the X- axis, the time is taken as the Y – axis and the time is measured in minutes. During the initial stage the performance of the existing is very low for processing. The stability and the progress are much higher in the proposed system.

IX. CONCLUSION

Transport Control Protocol (TCP) incast congestion happens when number of senders work in parallel with the same server where there is high bandwidth and low latency network problem occurs. Incast congestion degrades the performance as the packets are lost at server side due to buffer overflow and hence the response time will be more. To improve the performance the focus is on the TCP throughput, handling failures, scheduling and allocation. In this FSSP algorithm is implemented that adjust the TCP receive window proactively active before packet loss occurs by means of subserver allocation through grid computing. From the above graph we got the result that Packet losses are reduced with the prevention of congestion to the large extend and also the bandwidth is effectively utilized.

References

1. Zebo Feng , Xiaoping Wu , Liangli Ma ; Wei Ren, “Establishing the security foundations for Network protocol design,” in proceedings of Communication Technology (ICCT), IEEE 14th International Conference, pp. 789 – 793, ISBN: 978-1-4673-2100-6, Nov. 2012.
2. Hamlyn, A. , Cheung, H. , Mander, T. , Lin Wang , Cungang Yang , Cheung, R., “Network Security Management and Authentication of Actions for Smart Grids Operations,” in proceedings of Electrical Power Conference, 2007. EPC 2007. IEEE Canada, pp. 31 – 36, E-ISBN: 978-1-4244-1445-1, Oct. 2007.
3. J. Dean and S. Ghemawat, “MapReduce: Simplified data processing on large clusters,” in Proc. OSDI, vol. 53, no. 1, pp. 10, January 2004.
4. H. Wu, Z. Feng, C. Guo, and Y. Zhang, “ICTCP: Incast congestion control for TCP in data Center networks,” in Proc. Networking, IEEE/ACM Transactions, Vol. 21, no. 2, pp. 345-358, ISSN: 1063-6692, April 2013.
5. R. Prasad, M. Jain, and C. Dovrolis, “Socket buffer auto-sizing for high-performance data Transfers,” J. Grid Computer., vol. 1, no. 4, pp. 361–376, 2003.
6. P. Mehra, A. Zakhor, and C. Vl eeschouwer, “Receiver-driven bandwidth sharing for TCP,” in Proc. IEEE INFOCOM, 2003, vol. 2, pp.1145–1155.
7. N. Spring,M. Chesire,M. Berryman, and V. Sahasranaman, “Receiver based management of Low bandwidth access links,” in Proc. IEEE INFOCOM, 2000, vol. 1, pp. 245–254.
8. Y. Chen, R. Griffith, J. Liu, R. Katz, and A. Joseph, “Understanding TCP incast throughput Collapse in datacenter networks,” in Proc.WREN, 2009, pp. 73–82.
9. L. Brakmo and L. Peterson, “TCP Vegas: End to end congestion avoidance on a global Internet,” IEEE J. Sel. Areas Commun., vol. 13, no. 8, pp. 1465–1480, Oct. 1995.

10. M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network Architecture," in Proc. ACMSIGCOMM, 2008, pp. 63–74.
11. E. Krevat, V. Vasudevan, A. Phanishayee, D. Andersen, G. Ganger, G. Gibson, and S. Seshan, "On application-level approaches to avoiding TCP throughput collapse in cluster-Storage systems," in Proc. Supercomput., 2007, pp. 1–4.
12. A. Phanishayee, E. Krevat, V. Vasudevan, D. Andersen, G. Ganger, G.Gibson, and S. Seshan, "Measurement and analysis of TCP throughput collapse in cluster-based storage Systems," in Proc. USENIX FAST, 2008, Article no. 12.
13. C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang,and S. Lu, "BCube: A High performance, server-centric network architecture for modular data centers," in Proc. ACM SIGCOMM, 2009, pp.63–74.
14. C. Guo, H.Wu,K.Tan,L. Shi,Y.Zhang, and S. Lu, "DCell:Ascalable and fault tolerant Network structure for data centers," in Proc. ACM SIGCOMM, 2008, pp. 75–86.
15. D. Nagle, D. Serenyi, and A. Matthews, "The Panasas ActiveScale storage cluster: Delivering scalable high bandwidth storage," in Proc. SC, 2004, pp. 53.
16. M. Alizadeh, A. Greenberg, D.Maltz, J. Padhye, P. Patel, B.Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," in Proc. SIGCOMM, 2010, pp. 63–74.
17. D. B. J. Yih-Chun Hu, Adrian Perrig. Ariadne, "A secure on-demand routing protocol for adhoc networks," in Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking (MobiCom 2002)," Sept. 2002.
18. H Yang, H Y. Luo, F Ye, S W. Lu, and L Zhang, "Security in mobile ad hoc networks: Challenges and solutions" IEEE Wireless Communications.11 (1), pp. 38-47, 2004.

