CAP* Theorem vs. CAS** Principle in Computer Networks

Amina Y. AlSallut [#1], Hana H. Hejazi [#2], Heba A. AbuGhali [#3]

[#] *Computer Engineering Department, Islamic University*
Gaza, Palestine
[1]aminay2005@hotmail.com
[2]h_hejaze34@hotmail.com
[3]heba_ali_7@hotmail.com

***Abstract*— Previously no body talks as we know about the relation between scalability, availability and consistency in computer networks, although the scalability is important issue. In this paper we study the tradeoff between consistency, availability and scalability to see if it possible to meet all of them in the network at the same time, and we found that we can not keep them together simultaneously in the same network. We used two simulators NS2 and OPNET simulator to achieve this result.**

***Keywords*— Consistency, Availability, Partition tolerance, and Scalability.**

I. Introduction

The world became a small village, the need of scalable network is growing, but that network must be available and consistence to meet the customer requirements, so we study if we can acquire this network.

The CAP theorem states that in computing when it comes to consistency, availability and partition tolerance you can have only two of the three, so if we take consistency and availability, we forfeit partitions e.g. Single-site databases, anther case if we take consistency and partition tolerance, we forfeit availability e.g. Distributed databases and the last case if we take availability and partition tolerance, we forfeit consistency e.g. DNS.

The CAS principle states that in computing when it comes to Consistency, Availability and Scalability you can have only two of the three.

First we study the CAP theorem as shown in the related work (section two), and then we study two papers the first one about scalability with consistency and the other paper is about scalability with availability.

In section three we state CAS principle and prove it, we use NS2 simulator in getting the result which shown in section four, and finally section five contain the conclusion.

* CAP is referring to consistency, availability and partition-tolerance

** CAS is referring to consistency, availability and scalability.

II. Related Work

A. CAP theorem

Brewer and Fox explain CAP principle in [1] and propose two strategies to improve the over all availability, he said that CAP principle consistency, availability and partition resilience pick at most two, and inscribe some examples that illustrate it.

In [2] the authors proved CAP principle in asynchronous network model and they propose solution in partially synchronous model.

In this paper we find these theorems with their proof**:**

Theorem 1: it's impossible in asynchronous network model to implement a read/ write data object that guarantees the following properties: availability and consistency in all fair executions. (Including those in which message is lost)

Corollary 1.1: it's impossible in asynchronous network model to implement a read/ write data object that guarantees the following properties: availability in all fair executions and consistency in all fair executions in which no message are lost.

Theorem 2: it's impossible in partially synchronous network model to implement a read/ write data object that guarantees the following properties: availability and consistency in all fair executions (Including those in which message is lost).

Theorem 3: The modified centralized algorithm is t-Connected consistent

B. Consistency, Availability and Scalability

The CAS principle states that in computing when it comes to Consistency, Availability and Scalability you can have only two of the three.

The CAS principle: Consistency, Availability and Scalability, where scalability is a desirable property of a network, which indicates its ability to either handle growing amounts of work (large participating nodes)in a graceful manner, or to be readily enlarged

All studies take two parts either scalability with availability like in [4] or scalability with consistency just like in [3] they explore scalable consistency protocols that never require synchronization and communication between all nodes that have copies of related objects. They achieve this by developing a novel approach called local consistency (LC).

Where the old way to address scalability requirement of distributed services is to employ service replication and caching of service state at client nodes, thus, scalable implementations of a service lead to the object state being distributed across multiple nodes. Server replications and client caching introduce the problem of consistency among multiple copies of an object.

Here we want to study the three properties together, and see how the scalability affects on the availability and consistency in networks.

III. CAS PRINCIPLE

The CAS principle states that in computing when it comes to Consistency, Availability and Scalability you can have only two of the three.

CISCO discuss in [5] the scaling is one of most important factor in network outages as shown below

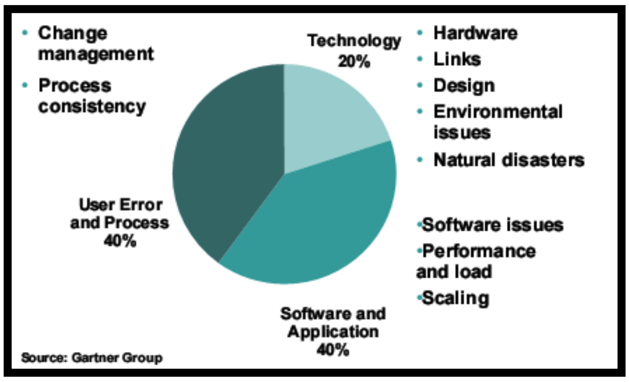


Fig. 1 cause of network outages.

To proof our principle we assume that we can get availability, consistency and we want to make the network scalable.

To make it consistent we need to add more data to the packet (checksum field) and we want to retransmit dropped packets, so the amount if data to be sent is increased.

In the other hand if the number of nodes is also increased due to scalability, congestion will occur and packets will be dropped (the node which send a request -and this request is dropped - will never get a replay from a receiver node), the receiver node will appear as unavailable.

First part: We conclude that to obtain consistency and scalability we must forfeit availability.

It can be illustrated in the following example

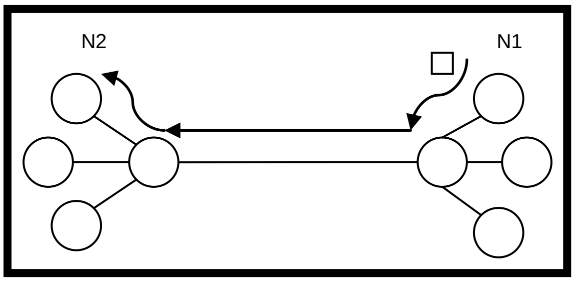


Fig. 2 A Network has a consistency property

In figure 2 the network works normally but if we try to increase the number of nodes in it, then we will be faced by the situation in figure3 due to increasing in packet size (consistency fields) and packet's number, so N2 will be unavailable (from N1 side of view); because it did not send a replay for the request packet which is dropped

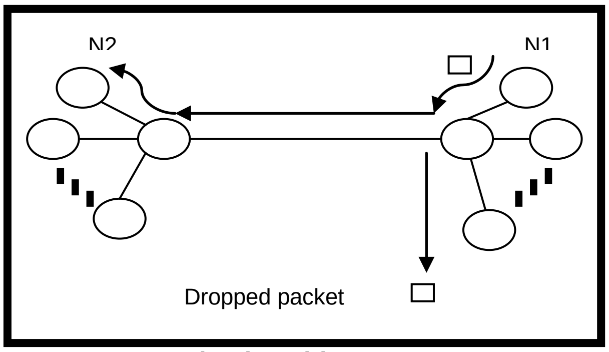


Fig. 3: packets drooped due to congestion.

The second part, if we want to make the network scalable and available, we need to redundant nodes (nodes that hold data to be sure that any request can be replayed), and if we update this data in one node we should update it in all node but this will increase the number of packets that move through a network.

Assume the network has a lot of nodes (due to scalability), a lot of packets (to get availability), congestion will be occurred and some packets will be dropped , so some node still has old version of data (we loss a consistency).

Second part: We conclude that to obtain availability and scalability we must forfeit consistency.

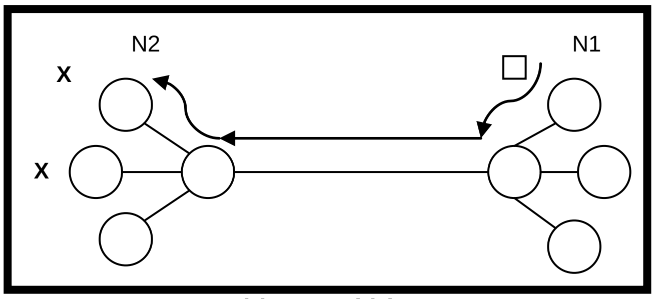


Fig. 4 A Network has an availability property.

In figure 4 the network works normally but if we try to increase the number of nodes in it, we will find the network as in figure 5 and the data is replicated in some nodes to in sure that one of them at least will be available to send replay for any request, as shown in figure 6 some node don't receive the packets and still have old version from data and we loss a consistency.

Now we want to make a network available and consistent, as shown previous to get availability you need redundancy, and to get consistency you need to increase amount of data to be sent (need checksum and update all version of redundant data) , so the network can not be too large to avoid congestion.

Third part: We conclude that to obtain availability and consistency we must forfeit scalability.

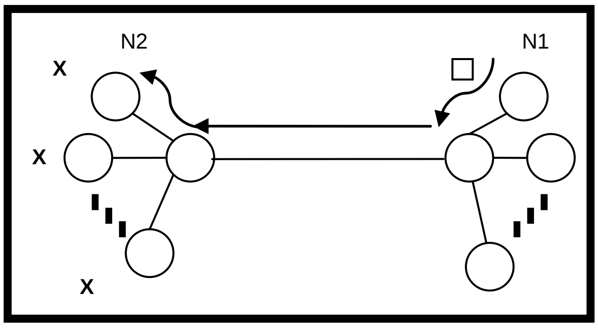


Fig. 5 the network after increase the node number.

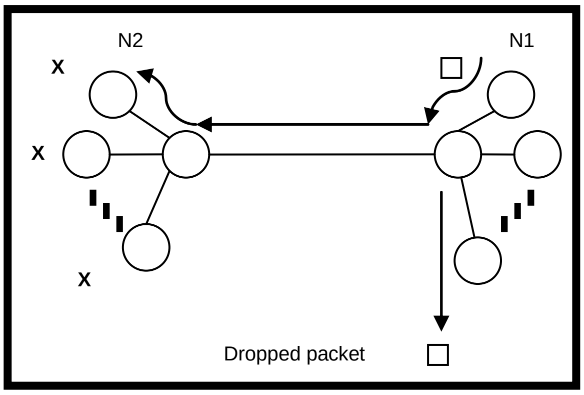


Fig. 6 packets drooped due to congestion.

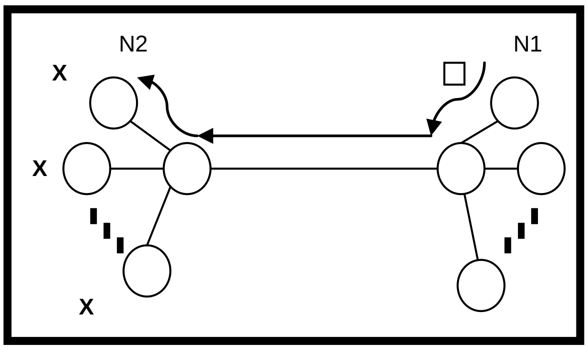


Fig.7 Network has an availability and consistency property.

In figure 7 the network works normally but if we try to increase the number of nodes in it, then we will be faced by the situation in figure 8 due to increasing in packet size (consistency fields) and packet's number due to redundancy (for availability).

So if the network growths, congestion will be happened.

IV. Simulation Result

We used NS2 simulator to test CAS principle, we worked into two trends:

First: study scalability and consistency with availability

We use the NW shown in fig. 9 start with n = 1 to n = 5, queue size is 10, and drop tail queue, having TCP and UDP connection on each nod the run time was 10.5 seconds .

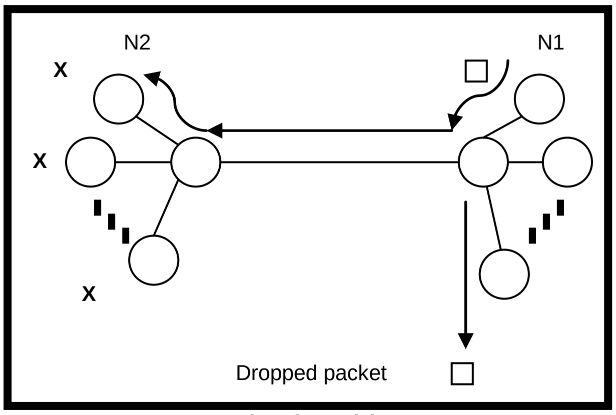


Fig. 8 packets drooped due to congestion.

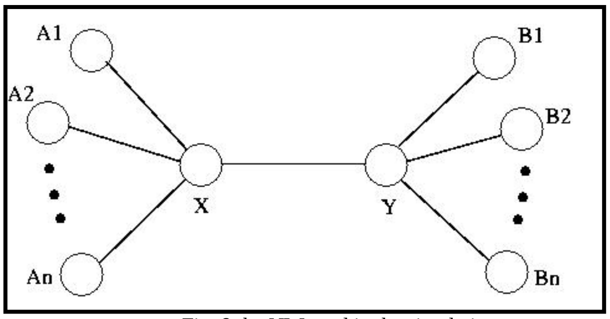


Fig. 9 the NW used in the simulation.

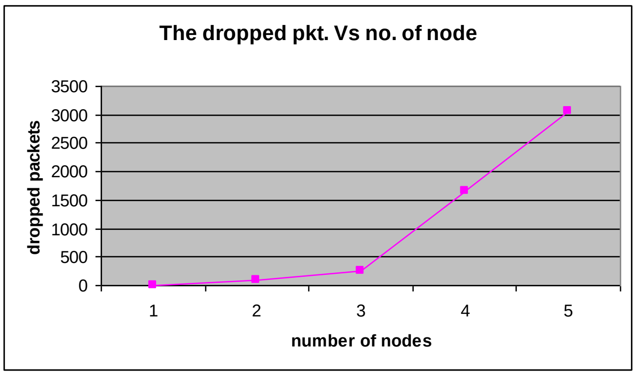


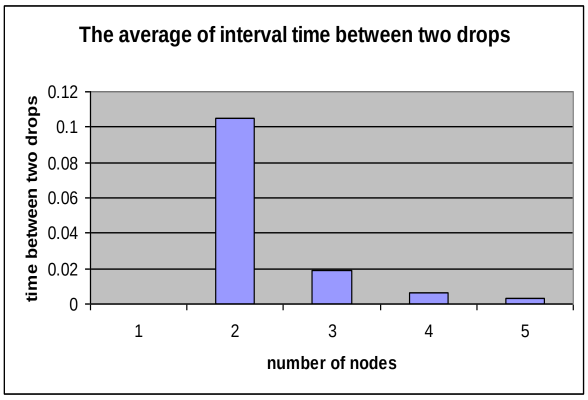
Fig. 10 The dropped packets Vs no. of node

As shown below in fig. 11. The average time between two drops is large, and get smaller while increasing the nodes, that's the no. dropped packets Increased.

Second: scalability and availability with consistency:

We use the same NW shown in fig. 12. A_1 to A_n is redundant from the data send from X to Y, the running time was 10.5. As notice in fig. 13. The same packets was send many times to node Y, without knowing which one is the up to date copy from the data requested

Fig. 11 The average of interval time between two drops



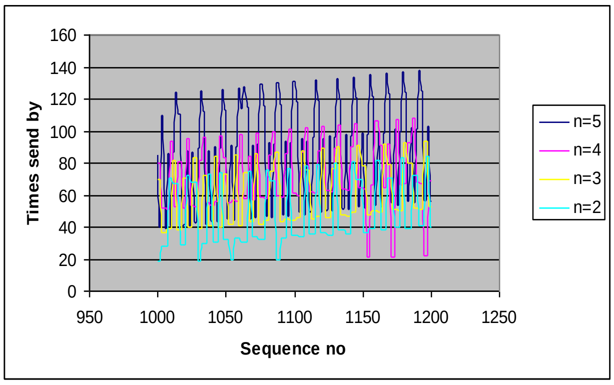


Fig. 13.b Redundant node = 2,3,4,5.

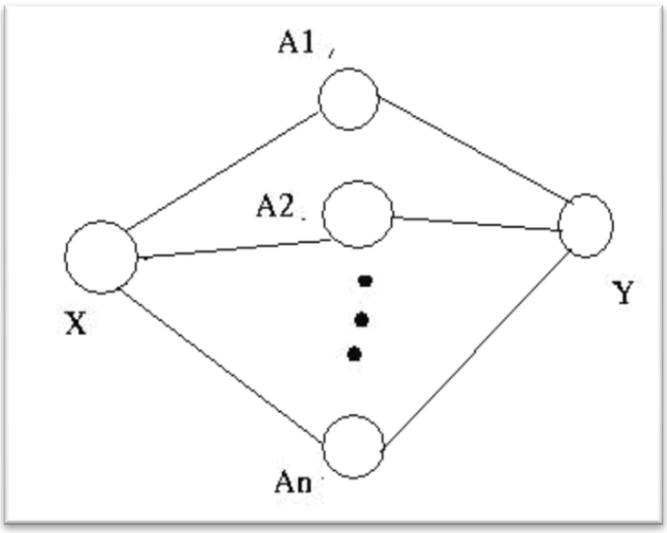


Fig. 12 The NW used in second part.

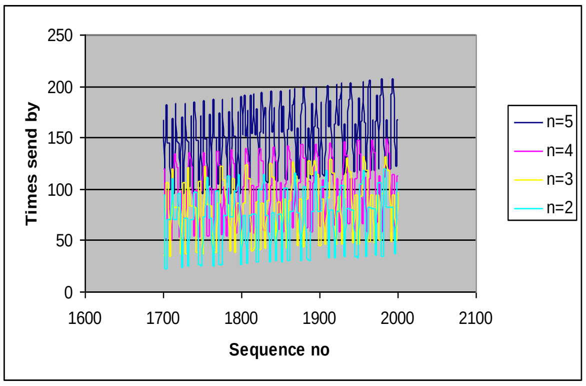


Fig. 14 redundant node = 2,3,4,5

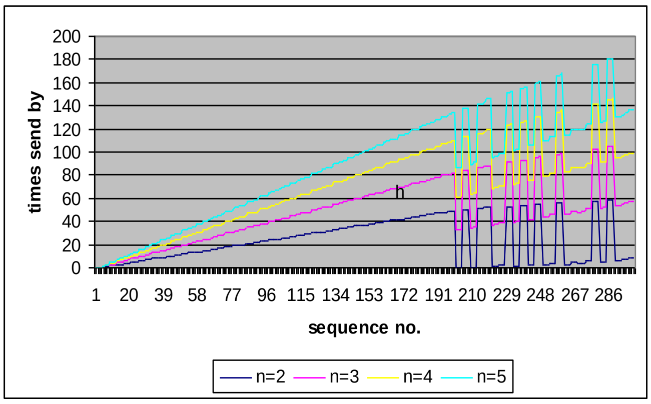


Fig. 13.a Redundant node = 2,3,4,5.

Figure 13.a, shows the sequence no 1 to 300, and figure 13.b, shows the sequence no 1000 to 1300. while figure 14, shows the sequence no 1700 to 2000 .From the tow figures, it's clear that due to increasing the no. of copies (increasing n), the destination get a lot of copies to the same data without knowing if being the last updated.

Third: availability and consistency with scalability

In this part we used OPNET simulator to test that if we ensure availability and consistency in the network, we can not put a lot nodes in the network, so we made two scenarios as shown below:

First scenario

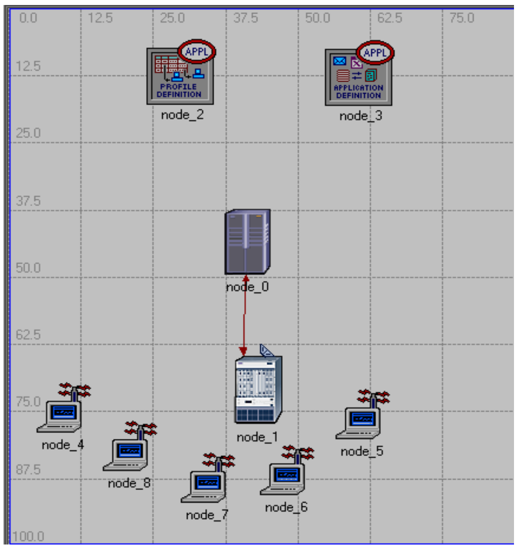


Fig. 15 First scenario

Second scenario

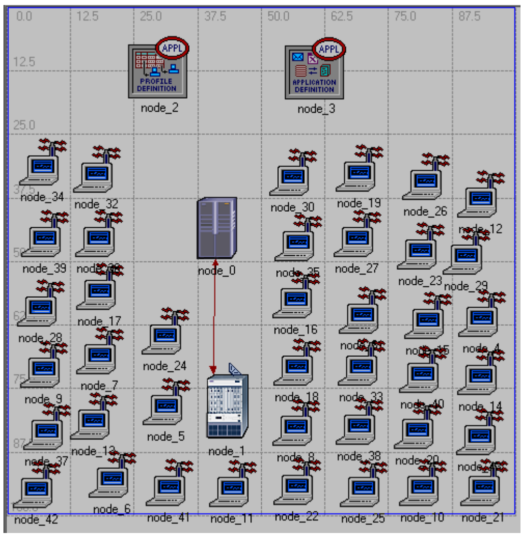


Fig. 16 Second scenario

In the first scenario we have 5 station but in the second one we have 40 station , and we found that the delay in the second scenario is greater than the first one which mean that there are dropped packets and retransmission which cause delay.

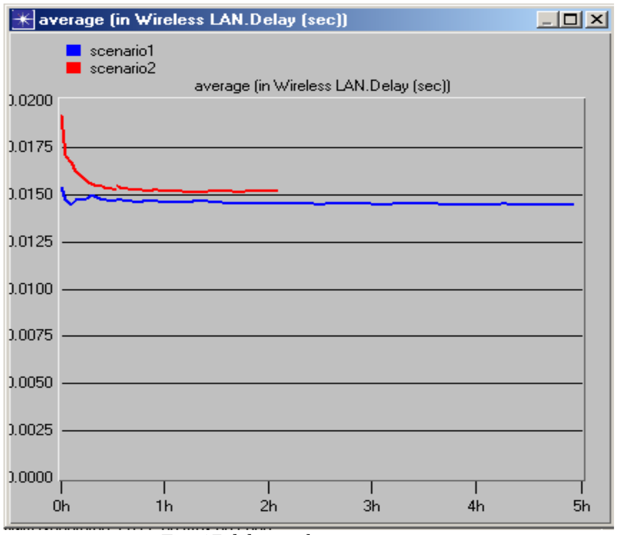


Fig. 17 delay in the two scenarios

V. Conclusion

Nowadays partitioning tolerance becomes unimportant for modern NW, but scalability comes to be in site as very important characteristic in modern NW, so we study CAS principle to show how can't getting the third together in the same NW in the same time. We show that by using ns2 simulator.

Acknowledgment

First we thank Dr. Mohammed Mikki to give us the chance to study this subject and his effort to encourage us during working in this paper, finally, we thank every one being attention to read this paper and give us any notice to progress this work.

References

[1] Armando Fox & Eric Brewer. Harvest, Yield, and Scalable Tolerant Systems. , Hot Topics in Operating Systems, IEEE, pages 174-178, 1999
[2] Seth Gilbert & Nancy Lynch .Brewer's Conjecture and the Feasibility of Consistent, Available, Partition – Tolerant Web Services ,ACM Pages: 51 - 59 Volume 33 , Issue 2 ,June 2002.
[3] Mustaque Ahamad & Rammohan Kordale. Scalable Consistency Protocols for Distributed Services, IEEE Transactions on Parallel and Distributed Systems archive, Volume 10, Issue 9, Pages: 888 - 903, September 1999
[4] Joel Snyder .Getting scalability, high availability from SSL VPN wares - Network World ,December 2005
[5] Cisco Systems, AVAILABILITY MEASUREMENT, SESSION NMS-2201, 9627_05_2004_c2, 2004 Cisco Systems, Inc.
[6] Hewlett-Packard Development Company, Consistency, availability, and partition-tolerance trade-offs on distributed data access systems, 5983-2544EN, 05/2005