A

Seminar report

On

RAIN Technology

Submitted in partial fulfillment of the requirement for the award of degree
Of Electronics

SUBMITTED TO:
www.studymafia.org

SUBMITTED BY:
www.studymafia.org

Preface

I have made this report file on the topic **RAIN Technology**; I have tried my best to elucidate all the relevant detail to the topic to be included in the report. While in the beginning I have tried to give a general view about this topic.

My efforts and wholehearted co-corporation of each and everyone has ended on a successful note. I express my sincere gratitude to ……..who assisting me throughout the preparation of this topic. I thank him for providing me the reinforcement, confidence and most importantly the track for the topic whenever I needed it.

Acknowledgement

I would like to thank respected Mr…….. and Mr. ……..for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present a seminar report. It helped me a lot to realize of what we study for.

Secondly, I would like to thank my parents who patiently helped me as i went through my work and helped to modify and eliminate some of the irrelevant or un-necessary stuffs.
Thirdly, I would like to thank my friends who helped me to make my work more organized and well-stacked till the end.

Next, I would thank Microsoft for developing such a wonderful tool like MS Word. It helped my work a lot to remain error-free.

Last but clearly not the least, I would thank The Almighty for giving me strength to complete my report on time.

ABSTRACT

The RAIN project is research collaboration between Caltech and NASA-JPL on distributed computing and data storage systems for future space-borne missions. The goal of the project is to identify and develop key building blocks for reliable distributed systems built with inexpensive off-the-shelf components.

The RAIN platform consists of a heterogeneous cluster of computing and/or storage nodes connected via multiple interfaces to networks configured in fault-tolerant topologies. The RAIN software components run in conjunction with operating system services and standard network protocols. Through software-implemented fault tolerance, the system tolerates multiple node, link, and switch failures, with no single point of failure.

The RAIN technology has been transferred to RAIN finity, a start-up company focusing on creating clustered solutions for improving the performance and availability of Internet data centers.

INTRODUCTION

RAIN technology originated in a research project at the California Institute of Technology (Caltech), in collaboration with NASA's Jet Propulsion Laboratory and the Defense Advanced Research Projects Agency (DARPA). The name of the original research project was RAIN, which stands for Reliable Array of Independent Nodes. The main purpose of the RAIN project was to identify key software building blocks for creating reliable distributed applications using off-the-shelf hardware. The focus of the research was on high-performance, fault-tolerant and portable clustering technology for space-borne computing. Led by Caltech professor Shuki Bruck, the RAIN research team in 1998 formed a company called Rainfinity. Rainfinity, located in Mountain View, Calif., is already shipping its first commercial software package derived from the RAIN technology, and company officials plan to release several other Internet-oriented applications. The RAIN project was started four years ago at Caltech to create an alternative to the expensive, special-purpose computer systems used in space missions. The Caltech Researchers wanted to put together a highly reliable and available computer system by distributing processing across many low-cost commercial hardware and software components.

To tie these components together, the researchers created RAIN software, which has three components:

1. A component that stores data across distributed processors and retrieves it even if some of the processors fail.
2. A communications component that creates a redundant network between multiple processors and supports a single, uniform way of connecting to any of the processors.
3. A computing component that automatically recovers and restarts applications if a processor fails.

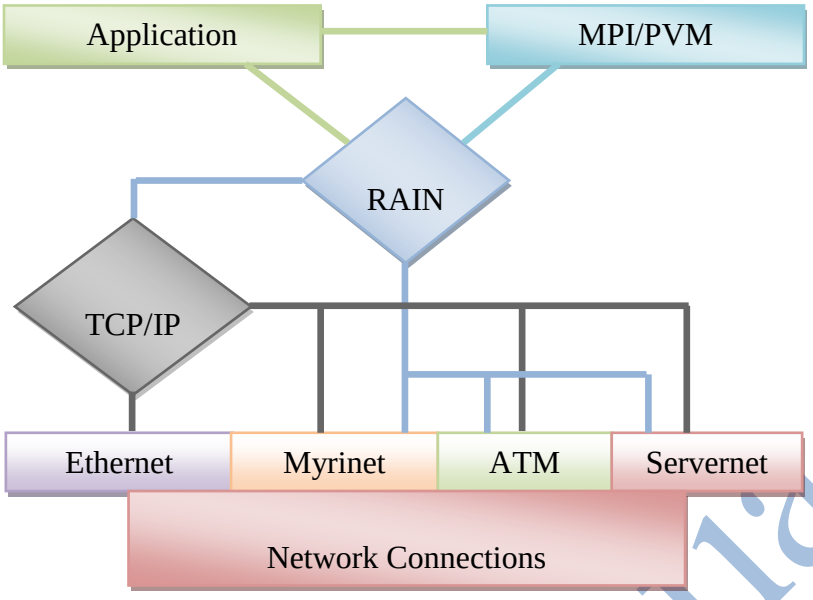


Figure1: RAIN Software Architecture

Myrinet switches provide the high speed cluster message passing network for passing messages between compute nodes and for I/O. The Myrinet switches have a few counters that can be accessed from an ethernet connection to the switch. These counters can be accessed to monitor the health of the connections, cables, etc. The following information refers to the 16-port, the clos-64 switches, and the Myrinet2000 switches.

ServerNet is a switched fabric communications link primarily used in proprietary computers made by Tandem Computers, Compaq, and HP. Its features include good scalability, clean fault containment, error detection and failover.

The ServerNet architecture specification defines a connection between nodes, either processor or high performance I/O nodes such as storage devices. Tandem Computers developed the original ServerNet architecture and protocols for use in its own proprietary computer systems starting in 1992, and released the first ServerNet systems in 1995.

Early attempts to license the technology and interface chips to other companies failed, due in part to a disconnect between the culture of selling complete hardware / software / middleware computer systems and that needed for selling and supporting chips and licensing technology.

A follow-on development effort ported the Virtual Interface Architecture to ServerNet with PCI interface boards connecting personal computers. Infiniband directly inherited many ServerNet features. After 25 years, systems still ship today based on the ServerNet architecture.

ORIGIN

1. Rain Technology developed by the California Institute of technology, in collaboration with NASA's Jet Propulsion laboratory and the DARPA.
2. The name of the original research project was RAIN, which stands for Reliable Array of Independent Nodes.
3. The RAIN research team in 1998 formed a company called Rainfinity.

ARCHITECTURE

The RAIN technology incorporates a number of unique innovations as its core modules:

Reliable transport ensures the reliable communication between the nodes in the cluster. This transport has a built-in acknowledgement scheme that ensures reliable packet delivery. It transparently uses all available network links to reach the destination.

When it fails to do so, it alerts the upper layer, therefore functioning as a failure detector. This module is portable to different computer platforms, operating systems and networking environments. Consistent global state sharing protocol provides consistent group membership, optimized information distribution and distributed group-decision making for a RAIN cluster.

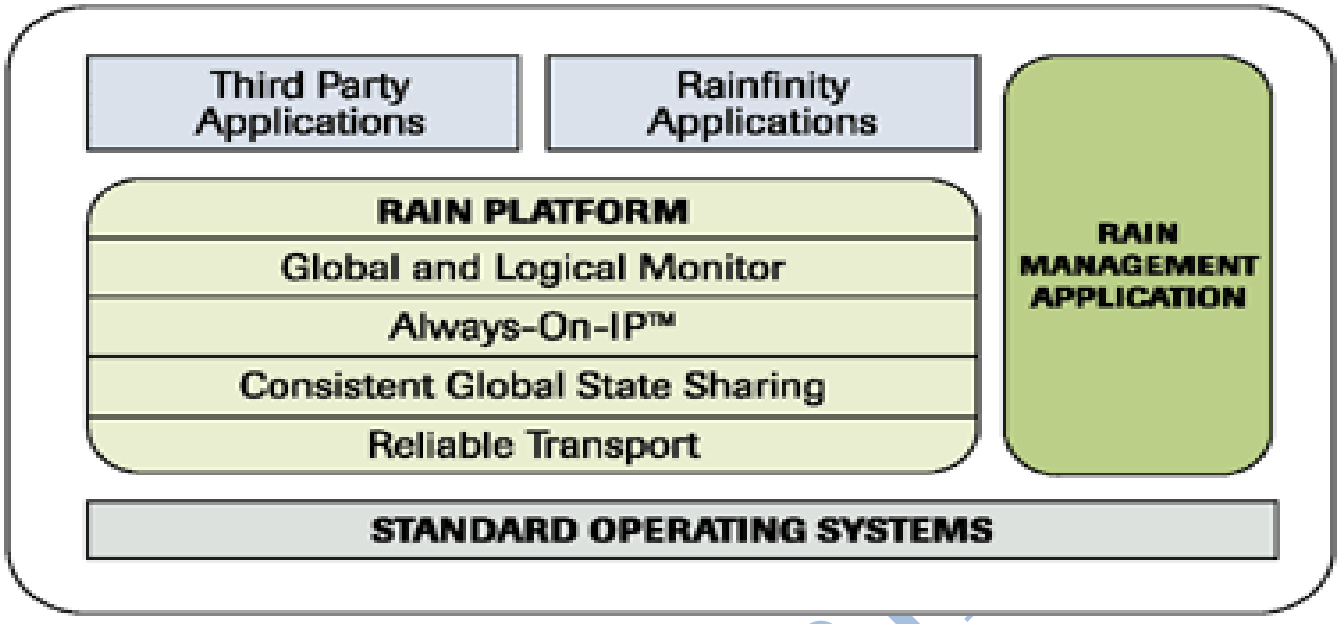
This module is at the core of a RAIN cluster. It enables efficient group communication among the computing nodes, and ensures that they operate together without conflict. Always on IP maintains pools of "always-available" virtual IPs.

This virtual IPs is nothing but the logical addresses that can move from one node to another for load sharing or fail-over. Usually a pool of virtual IPs is created for each subnet that the RAIN cluster is connected to. A pool can consist of one or more virtual IPs.

Always on IP guarantees that all virtual IP addresses representing the cluster are available as long as at least one node in the cluster is operational. In other words, when a physical node fails in the cluster, its virtual IP will be taken over by another healthy node in the cluster.

Local and global fault monitors monitor, on a continuous or event-driven basis, the critical resources within and around the cluster: network connections, Rainfinity or other applications residing on the nodes, remote nodes or applications.

It is an integral part of the RAIN technology, guaranteeing the healthy operation of the cluster.



FEATURES OF RAIN

1. **Communication.**
   - i) Bundled interface.
   - ii) Link monitoring.
   - iii) Fault-tolerant interconnects topologies.
     - ✓ The Problem
     - ✓ A Naïve Approach
     - ✓ Diameters Construction dc=2
2. **Data Storage.**
3. **Group Membership.**
   - ❖ Token Mechanism.
     - ➢ Aggressive Failure Detection.
     - ➢ Conservative Failure Detection.
     - ➢ Uniqueness of Tokens.
     - ➢ 911 Mechanisms
       - Token Regeneration
       - Dynamic Scalability
       - Link Failures and Transient Failures

1 - Communication

As the network is frequently a single point of failure, RAIN provides fault tolerance in the network through the following mechanisms:

i) **Bundled interfaces:** Nodes are permitted to have multiple interface cards. This not only adds fault tolerance to the network, but also gives improved bandwidth.

ii) **Link monitoring:** To correctly use multiple paths between nodes in the presence of faults, we have developed a link state monitoring protocol that provides a consistent history of the link state at each endpoint.

iii) **Fault-tolerant interconnects topologies:** Network partitioning is always a problem when a cluster of computers must act as a whole. We have designed network topologies that are resistant to partitioning as network elements fail.

- ✓ **The Problem**:

  We look at the following problem: Given n switches of degree ds connected in a ring, what is the best way to connect n compute nodes of degree dc to the switches to minimize the possibility of partitioning the compute nodes when switch failure occur? Figure3 illustrates the problem.

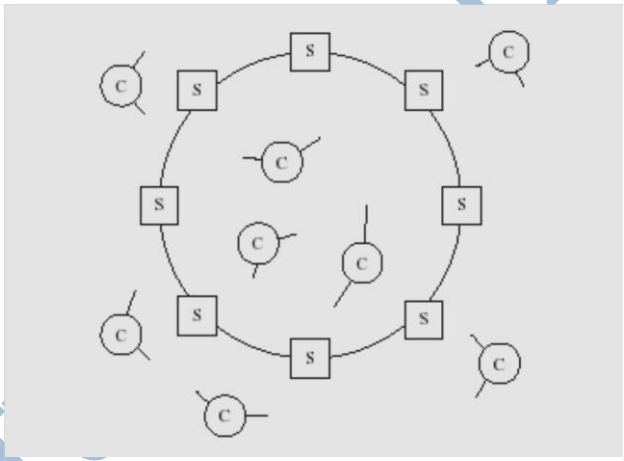


Figure 3

- ✓ **A Naïve Approach**:

  At a first glance, Figure 4a may seem a solution to our problem. In this construction we simply connect the compute nodes to the nearest switches in regular fashion. If we use this approach, we are relying entirely on fault tolerance in the switching network.

  A ring is 1-fault-tolerant for connectivity, so we can lose one switch without upset. A second switch failure can partition the switches and thus the compute nodes, as in figure 4b. this prompts the study of whether we can use the multiple connections of the compute nodes to make the compute nodes more resistant to partitioning. In other word, we want a construction where the

connectivity of the nodes is maintained even after the switch network has become partitioned.

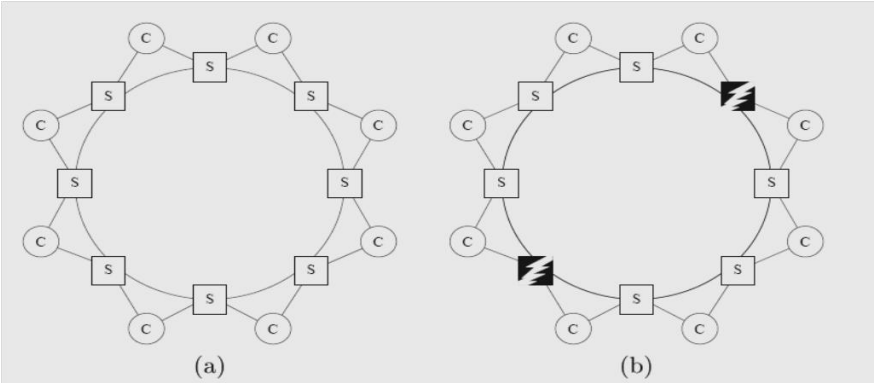


Figure 4: (a) A naïve approach, $d_c = 2$. (b) Notice that it is easily partitioned with two switch failures.

- ✓ **Diameters Construction dc=2;**

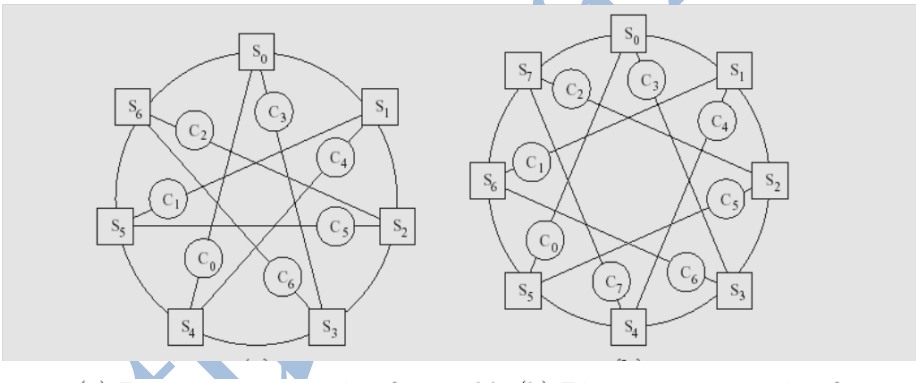


Figure 5: (a) Diameter construction for n odd. (b) Diameter construction for n even.

2 - Data Storage

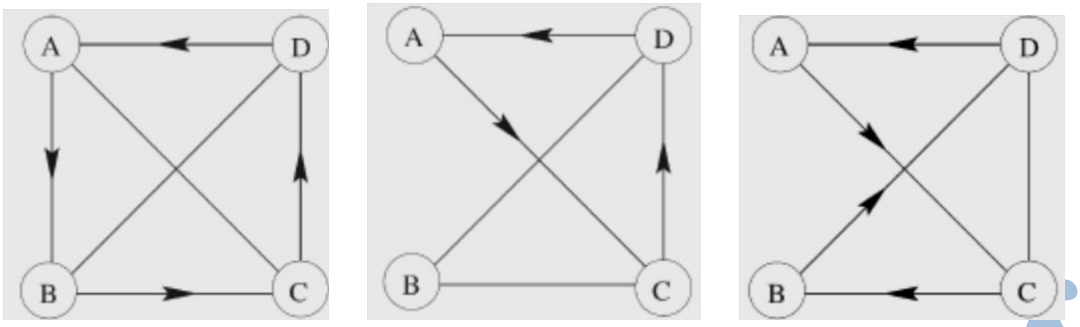
Fault tolerance in data storage over multiple disks is achieved through redundant storage schemes. Novel error-correcting codes have been developed for this purpose. These are array codes that encode and decode using simple XOR operations. Traditional RAID codes generally allow mirroring or parity as options. Array codes exhibit optimality in the storage requirements as well as in the number of update operations needed. Although some of the original motivations for these codes come from traditional RAID systems, these schemes apply equally well to partitioning data over disks on distinct nodes or even partitioning data over remote geographic locations.

3 - Group Membership

Tolerating faults in an asynchronous distributed system is a challenging task. Reliable group Membership service ensures that processes in a group maintain a consistent view of the global membership. In order for a distributed application to work correctly in the presence of faults, a certain level of problems in an asynchronous distributed system such as consensus, group membership, commit and atomic broadcast that have been extensively studied by researchers. In the RAIN system, the group membership protocol is the critical building block. It is a difficult task especially when change in membership occurs, either due to failures or voluntary joins and withdrawals. In fact under the classical asynchronous environment, the group membership problem has been proven impossible to solve in the presence of any failures. The underlying reason for the impossibility is that according to the classical definition of asynchronous environment, processes in the system share no common clock and there is no bound on the message delay. Under this definition it is impossible to implement a reliable fault detector, for no fault detector can distinguish between a crashed mode and a very slow mode. Since the establishment of this theoretic result researchers have been striving to circumvent this impossibility. Theorists have modified the specification while practitioners have built a number of real systems that achieve a level of reliability in their particular environment.

❖ **Token Mechanism**

The nodes in the membership are ordered in a logical ring. A token is a message that is being passed at a regular interval from one node to next node in the ring. The reliable packet communication layer is used for the transmission of the token, and guarantees that the token will eventually reach the destination. The token carries the authoritative knowledge of the membership when a node receives a token; it updates its local membership information according to the token. The token is also used for failure detection. There are two variants for failure detection protocol in this token mechanism. The aggressive detection protocol achieves fast detection time but is more prone to incorrect decisions viz, it may temporarily exclude a node only in the presence of link failures. The conservative detection protocol excludes a node only when its communication has failed from all nodes in the connected component. The conservative failure detection protocol has slower detection time than the other detection protocol.



(a) (b) (c)

Fig: (a) Token movement with no link failure. (b) Token movement with one link failure and aggressive failure detection. (c) Token movement with one link failure and conservative failure detection.

➢ **Aggressive Failure Detection**

When the aggressive failure detection protocol is used, after a node fails to send a token to the next node, the former node immediately decides that the latter node has failed or disconnected, and removes information and passes the token to the next live node in the ring. This protocol does not guarantee that all nodes in the connected component are included in the membership at all times. If a node looses a connection to part of the system because of link failure, it could be excluded from the membership. The excluded node will automatically rejoin the system, however, via the 911 mechanism, which will describe in the next section. For eg., for the situation in figure(b), the link between A and B is broken. After node A fails to send the token to node B, the aggressive failure detection protocol excludes node B from the membership. The ring changes from ABCD to ACD until node B rejoins the membership when the 911 mechanism is activated.

➢ **Conservative Failure Detection**

In comparison when conservative failure detection protocol is used, partially disconnected nodes will not be excluded. When a node detects that another node is not responding, the former node

does not remove the latter node from the membership instead it changes the order of the ring. In figure (c), after node A fails to send the token to node B, it changes the order of the ring from ABCD to ACBD. Node A then sends the token to node C, and C to node B. in the case when a node is indeed broken, all the nodes in the connected component fail to send the token to this node. When a node fails to send a token to another node twice in a row, it removes that node from the membership.

➢ **Uniqueness of Tokens**

The token mechanism is the basic component of the membership protocol. It guarantees that there exists no more than one token in the system at any time. This single token detects the failures, records the membership and updates all live nodes as it travels around the ring. After a failed node is determined, all live nodes in the membership are unambiguously informed within one round of token travel. Group membership consensus is therefore achieved.

➢ **911 Mechanisms**

Having described the token mechanism, few questions remain. What if a node fails when it processes the token and consequently the token is lost? Is it possible to add a new node to the system? How does the system recover from the transient failures? All of these questions can be answered by the 911 mechanism.

- **Token Regeneration**

To deal with the token loss problem, a time out has been set on each node in the membership. If a node does not receive a token for a certain period of time, it enters the STARVING mode. The node suspects that the token has been lost and sends out a 911 message to the next node in the ring. The 911 message is a request for a right to regenerate the token, and is to be provided by all the live nodes in the membership. It is imperative to allow one and only one node to regenerate the token when a token regeneration is needed. To guarantee this mutual exclusivity, we utilize the sequence number on the token.

Every time a token is being passed from one node to another, the sequence number on it is increased by one. The primary function of the sequence number is to allow the receiving node to

discard the out of sequence tokens. The sequence number also plays an important role in the token regeneration mechanism. Each node makes a local copy of the token every time that the node receives it. When a node needs to send a 911 message to request the regeneration of token, it adds this message to the sequence number that is on its last local copy of the token. This sequence number will be compared to all the sequence numbers on the local copies of the token on the other live nodes. The 911 requests will be denied by any node, which possesses a more recent copy of the token. In the event that the token is lost, every live node sends out a 911 request after its STARVING timeout expires. Only the node with the latest copy of the token will receive the right to regenerate the token.

- **Dynamic Scalability**

The 911 message is not only used as a token regeneration request, but also as a request to join the group. When a new node wishes to participate in the membership, it sends a 911 message to any node in the cluster. The receiving node notices that the originating node of this 911 is not a member of the distributed system, and therefore, treats it as a join request. The next time that it receives the token, it adds the new node to the membership, and sends the token to the new node. The new node becomes a part of the system.

- **Link Failures and Transient Failures**

The unification of the token regeneration request and the join request facilitates the treatment of the link failures in the aggressive failure detection protocol. Using the example in figure (b), node B has been removed from the membership because of the failure between A and B. node B does not receive the token for a while and it enters the STARVING mode and sends out a 911 message to node C. node C notices that node B is not a part of the membership and therefore treats the 911 as a join request. The ring is changed to ABCD and node B joins the membership.

Transient failures are treated with the same mechanism. When a transient failure occurs a node is removed from the membership. After the node recovers it sends out a 911 message. The 911 message is treated as a join request and the node is added back into the cluster. In the same

fashion, wrong decisions made in a local failure detector can be corrected, guaranteeing that all no faulty nodes in the primary connected component eventually stay in the primary membership.

Putting together the token and 911 mechanisms, we have a reliable group membership protocol. Using this protocol it is easy to build the fault management service. It is also possible to attach to the token application dependant synchronization information.

ADVANTAGES

1. RAIN Technology is the most scalable software cluster technology for the Internet market place today.
2. There is no limit on the size of a RAIN cluster.
3. All nodes are active and can participate in load balancing.
4. This software only technology is open and highly portable.

APPLICATIONS

We consider several applications implemented on RAIN platform based on the communication, fault management and data storage building blocks: a video server (RAIN Video), a web server (SNOW), and a distributed check pointing system (RAIN Check).

High availability video server:

There has been considerable research in the areas of fault tolerant internet and multimedia servers. Examples are the SunSCALR project at Sun Microsystems [15], For this RAIN Video application, a collection of videos are written and encoded to all n nodes in the system with distributed store operations. After this Each node runs a client application that attempts to display a video, as well as a server application that supplies encoded video data.

High availability web server:

SNOW is meant for Strong Network of Web Servers. It implements the concept project that demonstrates the features of the RAIN system. The main purpose is to develop a highly available

Fault-Tolerant Distributed Web Server Cluster that minimizes the risk of down time for mission critical Internet and intranet applications. The SNOW project uses several key building blocks of the RAIN technology. First, it considers the reliable communication layer is used to handle all of the messages, which passes between the servers in the SNOW system. Secondly, the token-based fault management module is used to establish the set of servers participating in the cluster.

Distributed check pointing mechanism:

A checkpoint and rollback/recovery mechanism on the RAIN platform based on the distributed store and retrieve operations. The scheme runs in conjunction with a leader election protocol. This protocol ensures that there is a unique node designated as leader in every connected set of nodes. As each job executes, a checkpoint of the state is taken periodically. The state is encoded and written to all accessible nodes with a distributed store operation. If a node fails or becomes inaccessible, the leader assigns the node's job to other nodes.

CONCLUSION

The goal of the RAIN project has been to address fault management, communication and storage in a distributed environment. Building blocks that we consider important are those providing reliable communication, group membership and reliable storage. Simply, RAIN allows for the grouping of an unlimited number of nodes, which can then function as one single giant node, sharing load or taking over if one or more of the nodes ceases to function correctly.

The future direction of this work is,

- Development of API's for using the various building blocks.
- The implementation of a real distributed file system

REFERANCES

[1]. www.wikipedia.com

[2]. www.searchdatacenter.techtarget.com

[3]. www.campusfever.com

[4]. www.google.com

[5]. www.seminartime.com

6 www.studymafia.org