International Journal On Engineering Technology and Sciences – IJETS™
ISSN(P): 2349-3968, ISSN (O): 2349-3976
Volume III, Issue V, May- 2016

INTEGRATING THE K-OUT-OF-N COMPUTING FRAMEWORK FOR ENERGY EFFICIENT AND FAULT TOLERANT IN MOBI-CLOUD

[1]Ms P.Sugeertha, [2] Mrs.P.Umarani,
Department of Computer Science and Engineering,
SSM College of Engineering, Komarapalayam,
Tamil Nadu, India.
[1] *Sugeertha8@gmail.com*, [2]*arunaiuma@gmail.com*

ABSTRACT

In Personal mobile devices have gained enormous popularity in recent years. Due to their limited resources (e.g., computation, memory, energy), however, executing sophisticated applications (e.g., video and image storage and processing, or map-reduce type) on mobile devices remains challenging because it need to maintain power consumption. Due to lacking of power in cloud computing of mobile devices, if any node becomes failure then, entire network of mobile communication may disordered. In this solution, mobile devices successfully retrieve or process data, in the most energy-efficient way, as long as k out of n remote servers are accessible. Through a real system implementation the method proves the feasibility of the proposed approach. Extensive simulations are demonstrated with fault tolerance and energy efficiency performance of the framework in larger scale networks The integrate k-out-of-n reliability mechanism into distributed computing in mobile cloud formed by only mobile devices. K-out-of-n, a well-studied topic in reliability control, ensures that a system of 'n' components operates correctly as long as k or more components work. More specifically, we investigate how to store data as well as process the stored data in mobile cloud with k-out of-n reliability such that: 1) The energy consumption for retrieving distributed data is minimized; 2) The energy consumption for processing the distributed data is minimized and 3) Data and processing are distributed considering dynamic topology changes.

Key word: Personal mobile devices, power consumption, fault tolerances, M-K-Out-Data Allocation, M-K-out-Data Processing

1.INTRODUCTION

Cloud computing is the delivery of computing services over the Internet. Cloud services allow individuals and businesses to use software and hardware that are managed by third parties at remote locations. Examples of cloud services include online file storage, social networking sites, web mail, and online business applications. The cloud computing model allows access to information and computer resources from anywhere that a network connection is available. Cloud computing provides a shared

pool of resources, including data storage space, networks, computer processing power, and specialized corporate and user applications.

Mobile Cloud Computing (MCC) is the combination of cloud computing, mobile computing and wireless networks to bring rich computational resources to mobile users, network operators, as well as cloud computing providers. The ultimate goal of MCC is to enable execution of rich mobile applications on a plethora of mobile devices, with a rich user experience. MCC provides business opportunities for mobile network operators as well as cloud providers.

In MCC, there are four types of cloud based resources, namely distant immobile clouds, proximate immobile computing entities, proximate mobile computing entities, and hybrid (combination of the other three models). Giant clouds such as Amazon EC2 are in the distant immobile groups whereas cloudlet or surrogates are member of proximate immobile computing entities. Smart phone's, tablets, handheld devices, and wearable computing devices are part of the third group of cloud-based resources which is proximate mobile computing entities.

In this research study the first framework to support fault-tolerant and energy-efficient remote storage and processing under a dynamic network topology, i.e., mobile cloud. The research framework aims for applications that require energy efficient and reliable distributed data storage and processing in dynamic network. They are integrating the k-out-of-n reliability mechanism into distributed computing in mobile cloud formed by only mobile devices. k-out-of-n, a well-studied topic in reliability control to ensures that a system of n components operates correctly as long as k or more components work. More specifically, they are study investigate how to store data as well as process the stored data in mobile cloud with k-out-of-n reliability

In paper our proposed study framework, a data object is encoded and partitioned into n fragments, and then stored on n different nodes. As long as k or more of the n nodes are available, the data object can be successfully recovered. Similarly, another set of n nodes are assigned tasks for processing the stored data and all tasks can be completed as long as k or more of the n processing nodes finish the assigned tasks. The parameters k and n determine the degree of reliability and different k; n pairs may be assigned to data storage and data processing. The following main contribution of the papers,

- Under utilization: Typical deployments have very low utilization of total computing capacity. Users want to run only a few applications per computer to obtain better response time. As a result, many computers are under-utilized.
- Security: Desktops are of ten managed by individual users and they have to regularly

apply security patches. Otherwise, the computers can become vulnerable.

- Operational costs: The total cost of ownership can grow rapidly for supporting increasing numbers of desktops and laptops, and for upgrading and up- dating software. Moreover, these computers may waste power as they are often kept on 24 h.
- The objective of this problem is to find n nodes in V as process n nodes such that energy consumption for processing a job of M tasks is minimized.

2.RELATED WORKS

In this paper, the authors explained that mobile applications are becoming increasingly ubiquitous and provide ever richer functionality on mobile devices. At the same time, such devices often enjoy strong connectivity with more powerful machines ranging from laptops and desktops to commercial clouds. This paper presents the design and implementation of CloneCloud, a system that automatically transforms mobile applications to benefit from the cloud. The system is a flexible application partitioner and execution runtime that enables unmodified mobile applications running in an application-level virtual machine to seamlessly off-load part of their execution from mobile devices onto device clones operating in a computational cloud.

In this paper, the authors explained that Smartphones have exploded in popularity in recent years, becoming ever more sophisticated and capable. As a result, developers worldwide are building increasingly complex applications that require ever increasing amounts of computational power and energy. They propose ThinkAir, a framework that makes it simple for developers to migrate their smartphone applications to the cloud. ThinkAir exploits the concept of smartphone virtualization in the cloud and provides method-level computation offloading. Advancing on previous work, it focuses on the elasticity and scalability of the cloud and enhances the power of mobile cloud computing by parallelizing method execution using multiple virtual machine (VM) images.
In this paper, they propose ThinkAir, a new mobile cloud computing framework which takes the best of the two worlds.ThinkAir addresses MAUI's lack of scalability by creating virtual machines (VMs) of a complete smartphone system on the cloud, and removes the restrictions on applications/inputs/environmental conditions that CloneCloud induces by adopting an online method-level offloading. Moreover, ThinkAir provides an efficient way to perform on-demand resource allocation, and exploits parallelism by dynamically creating, resuming, and destroying VMs in the cloud when needed.

In this paper [4] describe the mobile devices are increasingly being relied on for services that go beyond simple connectivity and require more complex processing. Fortunately, a mobile device encounters, possibly

intermittently, many entities capable of lending it computational resources. At one extreme is the traditional cloud-computing context where a mobile device is connected to remote cloud resources maintained by a service provider with which it has an established relationship. In this paper they consider the other extreme, where a mobile device's contacts are only with other mobile devices, where both the computation initiator and the remote computational resources are mobile, and where intermittent connectivity among these entities is the norm.

In this paper [5] describe the geospatially aware mobile devices, such as smart phones, rely on an architecture that is power con-strained and processing power limited. The utility of these devices can be increased by offloading compute-intensive applications to parallel high performance computing (HPC) architectures, thus limiting battery drain, allowing access to large data, and providing faster time to solution. Such a paradigm can be achieved through tactical cloudlets that must operate in environments dominated by mobile ad-hoc infrastructure (common in remote environments or military applications). Executing this paradigm is further complicated in that HPC nodes themselves (with some reduced mobility) are now deployable through the use of ruggedized hybrid core technologies. This paper discusses their concept for cloudlet seeding: the static strategic placement of HP Cassets in deployed settings in such a way to balance computational load and limit hops to both stationary and mobile HPC nodes.

2.1 EXISTING SYSTEM

The existing presents a mathematical model for both optimizing energy consumption and meeting the fault tolerance requirements of data storage and processing under a dynamic network topology. This paper presents an efficient algorithm for estimating the communication cost in a mobile cloud, where nodes fail or move, joining/leaving the network.

The project proposed a first process scheduling algorithm that is both fault-tolerant and energy efficient. It presents a distributed protocol for continually monitoring the network topology, without requiring additional packet transmissions. The evaluation of this proposed framework processed through a real hardware implementation and large scale simulations.

The framework, running on all mobile nodes, provides services to applications that aim to: (1) store data in mobile cloud reliably such that the energy consumption for retrieving the data is minimized (k-out-of-n data allocation problem); and (2) reliably process the stored data such that energy consumption for processing the data is minimized (k-out-of-n data processing problem).

DRAWBACKS

- If n-k node fails before the tasks completion, the task completion ratio is less than 1 since single cloud provider scenario is higher.
- Same execution cost is applied for both nodes with fixed power supply and nodes powered by battery.
- Multiple cloud provider scenarios with different execution cost for same task is not considered.
- Storage of node consideration becomes failure for data processing.
- Optimizing the energy efficient becomes tedious for calculating nodes availability.
- Failure of node with temporary and permanent disorder for calculating energy consumption.

2.2 PROPOSED SYSTEM

The proposed system includes all the existing system approach which covers multiple cloud service provider environments with different execution cost for same task. In general, each node may have different energy cost depending on their energy sources; e.g., nodes attached to a constant energy source may have zero energy cost while nodes powered by battery may have relatively high energy cost.

The applications generate data and our framework stores data in the network. For higher data reliability and availability, each data is encoded and partitioned into fragments; the fragments are distributed to a set of storage nodes. In order to process the data, applications provide functions that take the stored data as inputs.

Each function is instantiated as multiple tasks that process the data simultaneously on different nodes. Nodes executing tasks are processor nodes; we call a set of tasks instantiated from one function a job. Client nodes are the nodes requesting data allocation or processing operations. A node can have any combination of roles from: storage node, processor node, or client node, and any node can retrieve data from storage nodes. It considered Topology Discovery and Monitoring, Failure Probability Estimation, Expected Transmission Time (ETT) Computation, k-out-of-n Data Allocation and k-out-of-n Data Processing.

In general, each node may have different energy cost depending on their energy sources; e.g., nodes attached to a constant energy source may have zero energy cost while nodes powered by battery may have relatively high energy cost. For simplicity, we assume the network is homogeneous and nodes consume the same amount of energy for processing the same task. As a result ,only the transmission energy affects the energy efficiency of the final solution. We leave the modeling of the general case as future work

In existing system, it is assumed the network is homogeneous and nodes consume the same amount of energy for processing the

same task. As a result, only the transmission energy affects the energy efficiency of the final solution. But in proposed system, different execution cost for same task scenario is considered which resembles the real time case.

ADVANTAGES

- Even If n-k node fails before the tasks completion, the task completion ratio may be equal to 1 since multiple cloud provider scenario.
- Different execution cost is applied for both nodes with fixed power supply and nodes powered by battery.
- Multiple cloud provider scenarios with different execution cost for same task is considered.
- Improve a mathematical model for both optimizing energy consumption and meeting the fault tolerance requirements of data storage and processing under a dynamic network topology.
- An efficient algorithm for estimating the communication cost in a mobile cloud, where nodes fail or move, joining/leaving the network.
- Scheduling algorithm that is both fault-tolerant and energy efficient.
- A distributed protocol for continually monitoring the network topology, with requiring additional packet transmissions.

3. ENERGY ARCHITECTURE MODEL

The framework, running on all mobile nodes, provides services to applications that aim to:

- Store data in mobile cloud reliably such that the energy consumption for retrieving the data is minimized (k-out-of-n data allocation problem);
- Reliably process the stored data such that energy consumption for processing the data is minimized (k-out-of-n data processing problem).

The application running in a mobile ad-hoc network may generate a large amount of media files and these files must be stored reliably such that they are recoverable even if certain nodes fail. At later time, the application may make queries to files for information such as the number of times an object appears in a set of images. Without loss of generality, we assume a data object is stored once, but will be retrieved or accessed for processing multiple times later. They first define several terms. As shown in Fig. 1, applications generate data and our framework stores data in the network. For higher data reliability and availability, each data is encoded and partitioned into fragments; the fragments are distributed to a set of storage nodes. In order to process the data, applications provide functions that take the stored data as inputs. Each function is instantiated as multiple tasks that process the data simultaneously on different nodes. Nodes executing tasks are

processor nodes; we call a set of tasks instantiated from one function a job.

Client nodes are the nodes requesting data allocation or processing operations. A node can have any combination of roles from: storage node, processor node, or client node, and any node can retrieve data from storage nodes. As shown in Fig. 1, our framework consists of five components: Topology Discovery and Monitoring, Failure Probability Estimation, Expected Transmission Time (ETT) Computation, k-out-of-n Data Allocation and k-out-of-n Data Processing. When a request for data allocation or processing is received from applications, the Topology Discovery and Monitoring component provides network topology information and failure probabilities of nodes.

The failure probability is estimated by the Failure Probability component on each node. Based on the retrieved failure probabilities and network topology, the ETT Computation component computes the ETT matrix, which represents the expected energy consumption for communication between any pair of node. Given the ETT matrix, our framework finds the locations for storing fragments or executing tasks.

The k-out-of-n Data Storage component partitions data into n fragments by an erasure code algorithm and stores these fragments in the network such that the energy consumption for retrieving k fragments by any node is minimized. K is the minimal number of fragments required to recover a data. If an application needs to process the data, the k-outof-n Data Processing component creates a job of M tasks and schedules the tasks on n processor nodes such that the energy consumption for retrieving and processing these data is minimized. This component ensures that all tasks complete as long as k or more processor nodes finish their assigned tasks. The Topology Discovery and Monitoring component continuously monitors the network for any significant change of the network topology. It starts the Topology Discovery when necessary.

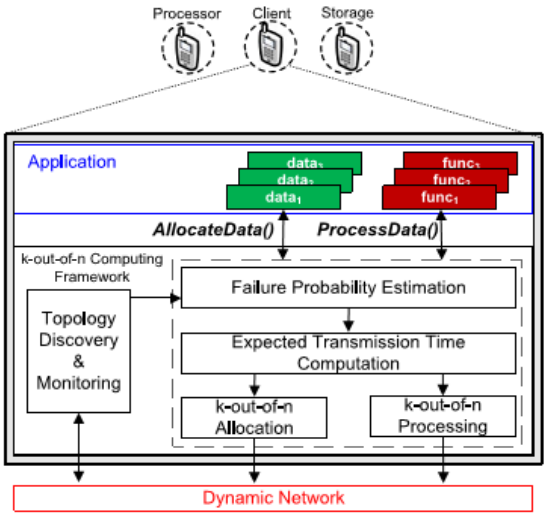


FIG. 1 Overview Of Our Proposed Framework Is Depicted

3.1 FORMULATION OF K-OUT-OF-N DATA ALLOCATION

In this problem, we are interested in finding n storage nodes denoted by S ¼ f g s1;

s2; . . . sn ; S V such that the total expected transmission cost from any node to its k closest storage nodes — in terms of ETT—is minimized. We formulate this problem as an ILP in Eqs. (1), (2), (3), (4), and (5).

$$R_{opt} = \arg\min_{R} \sum_{i=1}^{N} \sum_{j=1}^{N} D_{ij} R_{ij}$$

$$Subject\ to: \quad \sum_{j=1}^{N} X_j = n$$

$$\sum_{j=1}^{N} R_{ij} = k \ \forall i$$

$$X_j - R_{ij} \geq 0 \ \forall i$$

$$X_j \ and \ R_{ij} \in \{0, 1\} \ \forall i, j.$$

The first constraint (Eq. (2)) selects exactly n nodes as storage nodes; the second constraint (Eq. (3)) indicates that each node has access to k storage nodes; the third constraint (Eq (4)) ensures that jth column of R can have a non-zero element if only if Xj is 1; and constraints (Eq (5)) are binary requirements for the decision variables.

3.2 FORMULATION OF K-OUT-OF-N DATA PROCESSING

The objective of this problem is to find n nodes in V as processor nodes such that energy consumption for processing a job of M tasks is minimized. In addition, it ensures that the job can be completed as long as k or more processors nodes finish the assigned tasks. Before a client node starts processing a data object, assuming the correctness of erasure coding, it first needs to retrieve and decode k data fragments because nodes can only process the decoded plain data object, but not the encoded data fragment.

In general, each node may have different energy cost depending on their energy sources; e.g., nodes attached to a constant energy source may have zero energy cost while nodes powered by battery may have relatively high energy cost. For simplicity, we assume the network is homogeneous and nodes consume the same amount of energy for processing the same task. As a result, only the transmission energy affects the energy efficiency of the final solution. They leave the modeling of the general case as future work. Before formulating the problem, we define some functions: (1) f1(i) returns 1 if node i in S has at least one task; otherwise, it returns 0; (2) f2(j) returns the number of instances of task j in S; and (3) f3(z; j) returns the transmission cost of task j when it is scheduled for the zth time. We now formulate the k out of n data processing problem.

The objective function minimizes the total transmission cost for all processor nodes to retrieve their tasks. L represents the time slot of executing a task; i is the index of nodes in the network; j is the index of the task of a job. We note here that T r, the Data Retrieval Time Matrix, is a N M matrix, where the element T r ij corresponds to the estimated time for node i to retrieve task j. T r is computed by summing the transmission time (in terms of ETT available in

D) from node i to its k closest storage nodes of the task.

$$minimize \sum_{l=1}^{L} \sum_{i=1}^{N} \sum_{j=1}^{M} S_{lij} T_{ij}^{r}$$

$$Subject\ to: \quad \sum_{i}^{N} f_1(i) = n$$

$$f_2(j) = n - k + 1 \ \forall j$$

$$\sum_{l=1}^{L} S_{lij} \leq 1 \ \forall i, j$$

The first constraint ensures that n nodes in the network are selected as processor nodes. The second constraint indicates that each task is replicated k - 1 times in the schedule such that any subset of k processor nodes must contain at least one instance of each task. The third constraint states that each task is replicated at most once to each processor node. The fourth constraint ensures that no duplicate instances of a task execute at the same time on different nodes. The fifth constraint ensures that a set of all tasks completed at earlier time should consume lower energy than a set of all tasks completed at later time. In other words, if no processor node fails and each task completes at the earliest possible time, these tasks should consume the least energy.

$$\sum_{i=1}^{N} S_{lij} \leq 1 \ \forall l, j$$

$$\sum_{j=1}^{M} f_3(z_1, j) \leq \sum_{j=1}^{M} f_3(z_2, j) \ \forall z_1 \leq z_2.$$

3.3 K-OUT-OF-N DATA ALLOCATION

After the ETT matrix is computed, the k-out-of-n data allocation is solved by ILP solver. A simple example of how the ILP problem is formulated and solved is shown here. Considering Fig. 2b, distance Matrix D is a 4 4 symmetric matrix with each component Dij indicating the expected distance between node i and node j. Let's assume the expected transmissions times on all edges are equal to 1. As an example, D23 is calculated by finding the probability of two possible paths: 2 ! 1 ! 3 or 2 ! 4 ! 3. The probability of 2! 1 ! 3 is 0:8 0:8 0:9 0:4 ¼ 0:23 and the probability of 2! 4! 3 is 0:8 0:6 0:9 0:2 ¼ 0:08.

Another possible case is when all nodes survive and either path may be taken. This probability is 0:8 0:8 0:6 0:9 ¼ 0:34. The probability that no path exists between node 2 and node 3 is (1-0.23-0.08-0.34 ¼ 0.35). They assign the longest possible ETT ¼ 3, to the case when two nodes are disconnected. D23 is then calculated as 0:23 2 þ 0:08 2 þ 0:34 2 þ 0:35 3 ¼ 2:33. Once the ILP problem is solved, the binary variables X and R give the allocation of data fragments. In our solution, X shows that nodes 1-3 are selected as storage nodes; each row of R indicates where the client nodes should retrieve the data fragments from. For example, the first row of R shows that node 1 should retrieve data fragments from nodes 1 and 3.

$$D = \begin{pmatrix} 0.6 & 1.72 & 1.56 & 2.04 \\ 1.72 & 0.6 & 2.33 & 2.04 \\ 1.56 & 2.33 & 0.3 & 1.92 \\ 2.04 & 2.04 & 1.92 & 1.2 \end{pmatrix}$$

$$R = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix} \quad X = (1 \;\; 1 \;\; 1 \;\; 0).$$

3.4 K-OUT-OF-N DATA PROCESSING

The k-out-of-n data processing problem is solved in two stages—Task Allocation and Task Scheduling. In the Task Allocation stage, n nodes are selected as processor nodes; each processor node is assigned one or more tasks; each task is replicated to n k þ 1 different processor nodes. However, not all instances of a task will be executed—once an instance of the task completes, all other instances will be canceled.

The ask allocation can be formulated as an ILP .In the formulation, Rij is a N M matrix which predefines the relationship between processor nodes and tasks; each element Rij is a binary variable indicating whether task j is assigned to processor node i. X is a binary vector containing processor nodes, i.e., Xi ¼ 1 indicates that vi is a processor node.

The objective function minimizes the transmission time for n processor nodes to retrieve all their tasks. The first constraint indicates that n of the N nodes will be selected as processor nodes. The second constraint replicates each task to ðn k þ 1Þ different processor nodes. The third constraint ensures that the jth column of R can have a non-zero element if only if X j is 1; and the constraints are binary requirements for the decision variables.

Once processor nodes are determined, we proceed to the Task Scheduling stage. In this stage, the tasks assigned to each processor node are scheduled such that the energy and time for finishing at least M distinct tasks is minimized, meaning that we try to shorten t he job completion time while minimizing the overall energy consumption. The problem is solved in three steps.

$$\overline{R_{opt}} = \arg\min_{\overline{R}} \sum_{i=1}^{N} \sum_{j=1}^{M} T_{ij}^{r} \overline{R_{ij}}$$

$$\text{Subject to:} \quad \sum_{i=1}^{N} \overline{X_i} = n$$

$$\sum_{i=1}^{N} \overline{R_{ij}} = n - k + 1 \; \forall j$$

$$\overline{X_i} - \overline{R_{ij}} \geq 0 \; \forall i$$

$$\overline{X_j} \text{ and } \overline{R_{ij}} \in \{0, 1\} \forall i, j$$

First, they find the minimal energy for executing M distinct tasks in Rij. Second, they find a schedule with the minimal energy that h as the shortest completion time, tasks 1 to 3 are scheduled on different nodes at time slot 1; however, it is also possible that tasks 1 through 3 are allocated on the same node, but are scheduled in different time slots. These two steps are repeated n-k+1 times and M distinct tasks are scheduled upon each iteration. The third step is to shift tasks to earlier time slots.

A task can be moved to an earlier time slot as long as no duplicate task is running at the same time, task 1 on node 6 can be safely moved to time slot 2 because there is no task 1 scheduled at time slot 2. The ILP problem shown in Eqs. (17), (18), (19), and (20) finds M unique tasks from Rij that have the minimal transmission cost. The decision variable W is an N M matrix where R ij ¼ 1 indicates that task j is selected to be executed on processor node i.

The first constraint ensures that each task is scheduled exactly one time. The second constraint indicates that Wij can be set only if task j is allocated to node i in Rij. The last constraint is a binary requirement for decision matrix W.

The objective function minimizes integer variable Y, which is the largest number of tasks on one node. W_{ij} is a decision variable similar to W_{ij} defined previously. The first constraint ensures that the schedule cannot consume more energy that the E_{min} calculated previously. The second constraint schedules each task exactly once. The third constraint forces Y to be the largest number of tasks on one node. The last constraint is a binary requirement for decision matrix W.Once tasks are scheduled, we then rearrange tasks—tasks are moved to earlier time slots as long as there is free time slot and no same task is executed on other node simultaneously. Algorithm 1 depicts the procedure

$$W_E = \arg\min_{W} \sum_{i=1}^{N}\sum_{j=1}^{M} T_{ij}\overline{R_{ij}}W_{ij}$$

$$\text{Subject to:} \quad \sum_{i=1}^{N} W_{ij} = 1 \ \forall j$$

$$\overline{R_{ij}} - W_{ij} \geq 0 \ \forall i,j$$

$$W_{ij} \in \{0,1\} \ \forall i,j$$

$$\text{minimize } Y$$

$$\text{Subject to:} \sum_{i=1}^{N}\sum_{j=1}^{M} T_{ij} \times \overline{R_{ij}} \times \overline{W_{ij}} \leq E_{min}$$

$$\sum_{i=1}^{N} \overline{W_{ij}} = 1 \quad \forall j$$

$$\overline{R_{ij}} - \overline{W_{ij}} \geq 0 \ \forall i,j$$

$$Y - \sum_{j=1}^{M} \overline{W_{ij}} \geq 0 \quad \forall i$$

$$\overline{W_{ij}} \in \{0,1\} \quad \forall i,j$$

ALGORITHM : SCHEDULE RE-ARRANGEMENT

L ¼ last time slot in the schedule for time t ¼ 2! L do for each scheduled task J in time t do n processor node of task J while n is idle at t 1 AND J is NOT scheduled on any node at t 1 do Move J from t to t 1t ¼ t 1 end while end for end for

TOPOLOGY MONITORING

The Topology Monitoring component monitors the network topology continuously and runs in distributed manner on all nodes. Whenever a client node needs to create a file, the Topology Monitoring component provides the client with the most recent topology information immediately. When there is a significant topology change, it notifies the framework to update the current solution. We first give several notations. A term s refers to a state of a node, which can be either U or NU. The state becomes

U when a node finds that its neighbor table has drastically changed; otherwise, a node keeps the state as NU.

They let p be the number of entries in the neighbor table that has changed. A set ID contains the node IDs with p greater than t1, a threshold parameter for a "significant" local topology change. The Topology Monitoring component is simple yet energy-efficient as it does not incur significant communication overhead—it simply piggybacks node ID on a beacon message. The protocol is depicted in Algorithm 2. We predefine one node as a topology delegate Vdel who is responsible for maintaining the global topology information. If p of a node is greater than the threshold t1, the node changes its state to U and piggybacks its ID on a beacon message.

Whenever a node with state U finds that its p becomes smaller than t1, it changes its state back to NU and puts ID in a beacon message. Upon receiving a beacon message, nodes check the IDs in it. For each ID, nodes add the ID to set ID if the ID is positive; otherwise, remove the ID. If a client node finds that the size of set ID becomes greater than t2, a threshold for "significant" global topology change, the node notifies Vdel; and Vdel executes the Topology Discovery protocol. To reduce the amount of traffic, client nodes request the global topology from Vdel, instead of running the topology discovery by themselves. After Vdel completes the topology update, all nodes reset their status variables back to NU and set p ¼ 0.

ALGORITHM : DISTRIBUTED TOPOLOGY MONITORING

At each beacon interval:
if p > t1 and s 6¼ U then
s U
Put þ ID to a beacon message.
end if
if p t1 and s ¼ U then
s NU
Put ID to a beacon message.
end if
Upon receiving a beacon message on Vi:
for each ID in the received beacon message do
if ID > 0 then
ID ID S fIDg:
else
ID ID n fIDg:
end if
end for
if j j f g ID > t2 then
Notify Vdel and Vdel initiate topology discovery
end if
Add the ID in V 0
s beacon message.

RESULT AND DISCUSSION

4. EXPERIMENTAL RESULS

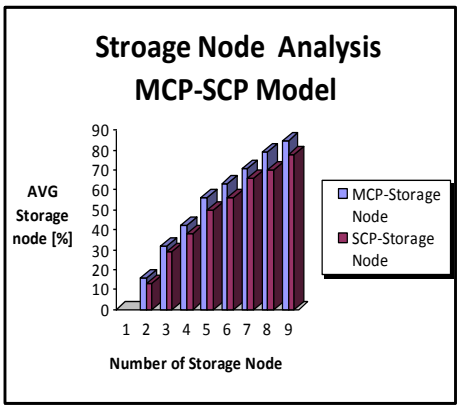
The **Table 7.1** represents experimental result for Single K-Out-N-Algorithm and M-K-Out-N-Algorithm model. The table contains

finding Data storage node count in K-Out-N and M-K-Out-N- Algorithm model within 1000 sec time interval details as shown.

S.NO	Number of Data Storage node (Count)	M-K-OUT-N-ALGORITHM		K-OUT-N-ALGORITHM	
		MCP - Storage Node	Selection of Storage Node	SCP-Storage Node	Selection of Storage Node
1	100	16	22	13	17
2	200	32	35	29	30
3	300	42	49	38	44
4	400	56	62	50	56
5	500	63	72	56	65
6	600	71	83	66	80
7	700	79	89	70	85
8	800	85	92	78	86

Table 7.1 Selection of Storage Node Analysis

The **Figure 7.1** represents experimental result for Single K-Out-N-Algorithm and M-K-Out-N-Algorithm model. The figure contains finding Data storage node count in K-Out-N and M-K-Out-N- Algorithm model within 1000 sce time interval details as shown.



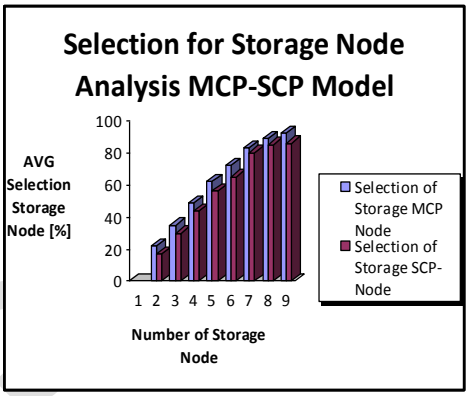


Figure 4.1 Selection Storage Node MEP-SCP Model

The **Table 4.2** represents experimental result for Single K-Out-N-Algorithm and M-K-Out-N-Algorithm mode time analysis. The table contains Data storage node allocation time analysis for K-Out-N and M-K-Out-N-Algorithm model within 1000 sec time interval details as shown.

S. No	Data Storage Node [n]	K-OUT-N [sec]	M-K-OUT-N [sec]
1	30	20	25

2	40	30	35
3	50	40	45
4	60	50	55
5	70	60	65
6	80	70	75
7	90	80	85
8	100	90	95
9	110	100	105
10	120	110	116

Table 4.2 Data storage Node- Time Analysis

The **Fig 4.2** represents experimental result for Single K-Out-N-Algorithm and M-K-Out-N-Algorithm mode time analysis. The figure contains Data storage node allocation time analysis for K-Out-N and M-K-Out-N-Algorithm model within 1000 sec time interval details as shown.

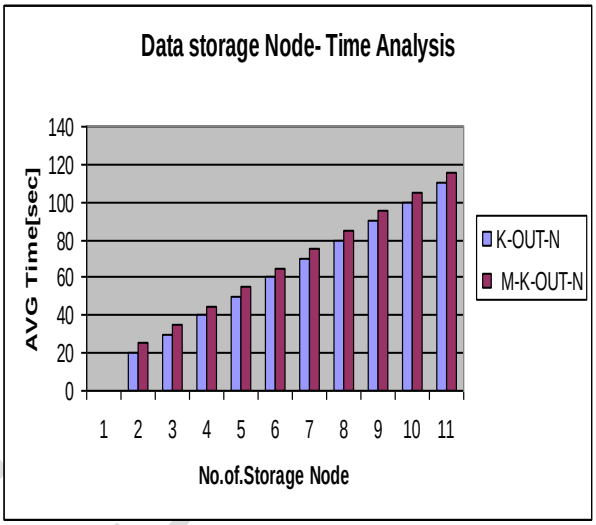


Fig 4.2 Data storage Node- Time Analysis

CONCLUSION AND FUTURE ENHANCEMENTS

CONCLUSION

In this thesis work presented the first k-out-of-n framework that jointly addresses the energy-efficiency and fault-tolerance problem overcome. It assigns data fragments to nodes such that other nodes retrieve data reliably with minimal energy consumption. It also allows nodes to process distributed data such that the energy consumption for processing the data is minimized. Through system implementation, the feasibility of our solution on real hardware was validated. Extensive simulations in larger scale networks proved the effectiveness of this thesis solution.

In this solution, mobile devices successfully retrieve or process data, in the most energy-efficient way, as long as k out of n remote servers are accessible. Through a real system implementation the method proves the feasibility of the proposed approach. Extensive simulations are demonstrated with fault tolerance and energy efficiency performance of the framework in larger scale networks with multi cloud provider.

FUTURE ENHANCEMENTS

In the future, to utilize the inferred information and extend the framework for efficient and effective Cloud services

monitoring and application design. The new system become useful if the below enhancements are made in future.

- The application can be web service oriented so that it can be further developed in any platform.
- The application if developed as web site can be used from anywhere.
- The algorithm can be further improved so that cost of the path can be further reduced
- The energy updation automatically reconfigured the storage node
- Currently the scheme has a slightly less memory overhead, while in the more complex applications; the scheme may utilize more memory. The future study can be in the area of more significant memory savings.

The new system is designed such that those enhancements can be integrated with current modules easily with less integration work. The new system becomes useful if the above enhancements are made in future. The new system is designed such that those enhancements can be integrated with current modules easily with less integration work.

REFERENCES

[1]. [Cuervo 2010] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl. MAUI: Making smart phones last longer with code offload. In MobiSys, 2010.

[2]. [Tolia 2006] N. Tolia, M. Kaminsky, D. G. Andersen, and S. Patil. An architecture for Internet data transfer. In NSDI, 2006.

[3]. L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. Mao, and L. Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In Proc. Int. Conf. Hardware/Software Codesign and System Synthesis, 2010.

[4]. A. Arora et al., "ExScal: Elements of an Extreme Scale Wireless Sensor Network," Proc. 11th IEEE Int'l. Conf. Embedded Real-Time Comp. Sys. Apps., Aug. 2005, pp. 102–8.

[5].T. He et al., "VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance," ACM Trans. Sensor Net., vol. 2, no. 1, 2006, pp. 1–38.

[6].A. Wood et al., "Context-Aware Wireless Sensor Networks for Assisted Living and Residential Monitoring," IEEE Network, vol. 22, no. 4, July 2008, pp. 26–33.

[7]. Dept. Homeland Security, National Incident Management System, Mar. 2004.

[8]. A. Jiang, BNetwork coding for joint storage and transmission with minimum cost,[in Proc. Int. Symp. Inf. Theory, Jul. 2006, pp. 1359–1363.

[9]. S.-Y. R. Li, R. W. Yeung, and N. Cai, BLinear network coding,[IEEE Trans. Inf. Theory, vol. 49, no. 2, pp. 371–381, Feb. 2003.

[10]. A. G. Dimakis, P. G. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran,

BNetwork coding for distributed storage systems,[IEEE Trans. Inf. Theory, vol. 56, no. 9, pp. 4539–4551, Sep. 2010.

[11]. Y. Wu, A. G. Dimakis, and K. Ramchandran, BDeterministic regenerating codes for
distributed storage,[in Proc. Allerton Conf. Control Comput. Commun., Monticello, IL, Oct. 2007.

[12]. R. Dougherty, C. Freiling, and K. Zeger, BInsufficiency of linear coding in network
information flow,[IEEE Trans. Inf. Theory, vol. 51, no. 8, pp. 2745–2759,
Aug. 2005.

[13]. [33] K. V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, BExact regenerating codes for distributed storage,[in Proc. Allerton Conf. Control Comput. Commun., Urbana, IL,
Sep. 2009.

[14]. C. Suh and K. Ramchandran, BExact regeneration codes for distributed storage repair using interference alignment,[in Proc. IEEE Int. Symp. Inf. Theory, Jun. 2010. [Online]. Available: http://arxiv.org/abs/ 1001.0107v2.

[15]. V. Cadambe, S. Jafar, and H. Maleki, BDistributed data storage with minimum storage regenerating codesVExact and functional repair are asymptotically equally efficient,[in Proc. IEEE Int. Workshop Wireless Netw. Coding, Apr. 2010. [Online]. Available:
http://arxiv.org/abs/1004.4299.

[16]. J. Li, S. Yang, X. Wang, and B. Li, BTree-structured data regeneration in distributed storage systems with regenerating codes,[in Proc. IEEE

[17]. S. Pawar, S. El Rouayheb, and K. Ramchandran, BOn security for distributed storage systems,[in Proc. IEEE Int. Symp. Inf. Theory, Jun. 2010

[18]. cloudstack - open source cloud computing. http://www.cloudstack.org/

[19]. IETF ALTO working group. http://datatracker.ietf.org/wg/alto/.

[20]. Windows Remote Desktop Services. http://tiny.cc/zt05bw.

[21]. M. D. Kristensen, "Execution plans for cyber foraging," in Proceedings of the 1st workshop on Mobile middleware: embracing the personal communication device, ser. MobMid '08. New York, NY, USA: ACM, 2008, pp. 2:1–2:6. [Online]. Available: http://doi.acm.org/10.1145/1462689.1462692

[22]. G. F. Carey, E. Barragy, R. McIay, and M. Sharma, "Element-by- element vector and parallel computations," Comm. Appl. Num. Methods, no. 4, pp. 299–308, 1988

[23] Z. Han, A. Swindlehurst, and K. Liu, "Smart deployment/movement of unmanned air vehicle to improve connectivity in manet," in Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE, vol. 1, april 2006, pp. 252 –257.

[24] R. Hunjet, A. Coyle, and M. Sorell, "Enhancing mobile adhoc networks through node placement and topology control," in Wireless Communication Systems (ISWCS), 2010 7th International Symposium on, sept. 2010, pp. 536 –540.

[25]. G. Karypis, "Multi-constraint mesh partitioning for contact/impact computations," in Proceedings of the 2003 ACM/IEEE conference on Supercomputing, ser. SC '03. New York, NY, USA: ACM, 2003, pp. 56–. [Online]. Available: http://doi.acm.org/10.1145/1048935.1050206

[26]. J. Huang, Q. Xu, B. Tiwana, Z. M. Mao, M. Zhang, and P. Bahl. Anatomizing Application Performance Differences on Smartphones. In Proc. of the 8th International Conference on Mobile Systems, Applications, and Services (MobiSys), San Francisco, CA, June 2010.