# Fault Tolerance Communication in Mobile Distributed Networks

**D. Bhuvana Suganth and R. Manjunath**

**Abstract** In mobile distributed networks, there may be possibility of unreliable communications and lack of flexibility in storing the data. In this paper, authors have proposed a distributed-based caching technique and error-free communication in mobile distributed networks. For every mobile host of the multicast tree, a mobile is placed to monitor the failure and to initiate failure recovery technique. Then the data storage is performed by selecting the caching policy based on game theory. The simulation result shows that this proposed technique enhances the reliability and reduces the communication failures.

**Keywords** Mobile distributed networks · Content distribution · Mobile host · Reliability

## 1 Introduction

Routing in mobile networks generally involves multiple hops, whereas in distributed networks, its a challenging task [1]. Distributed computing systems can provide several advantages, such as scalability, fault tolerance, and load balancing. Dealing with distributed systems and data storage together, introduces several challenges and difficulties [2], such as dynamic load balancing, unstable connections, communication failures, lack of flexibility in storing data, lack of auto-reconfigurations, limited radio range.

D. Bhuvana Suganth (✉) · R. Manjunath
AMC Engineering College, Bangalore, Karnataka, India
e-mail: bhuvanasuganthi@gmail.com

R. Manjunath
e-mail: manju_r99@yahoo.com



S.C. Satapathy et al. (eds.), *Proceedings of the International Conference on Data Engineering and Communication Technology*, Advances in Intelligent Systems and Computing 468, DOI 10.1007/978-981-10-1675-2_9

### 1.1 Reliability in Mobile Distributed Networks

A network's reliability has always been a major concern. Among different other factors such as software extensibility, maintainability, and usability, etc., Reliability have greater impact on software's life, because it can make the running application out of order [1, 2]. Reliability is a very broad term and any software application running in distributed environment can have various definitions.

A Software application is reliable if it can

- Performs well in specified time *t*.
- Do exactly the way it is designed as per requirements.
- Resist various failures and recover in case of any failure that occurs during system execution without producing any incorrect result.

The major key issues for ensuring reliability [3] is considered as

- errors
- inconsistency
- denial of services.

### 1.2 Problem Identification

The Failure detection is not as much efficient from [4, 5]. Failure recovery is not well organized from [5, 6]. The protocol used for multicasting is not having a considerable reliability from [7]. By considering all these problems, authors propose a technique called distributed caching and fault-tolerant communication in mobile distributed networks.

## 2 Methodology

A distributed mobile agent is deployed in each MH of the multicast tree [7] which performs the following tasks:

- Monitoring the failures of multicast tree members [4]
- Recovering from failures [4]
- Selecting the caching policy for content distribution based on Game theory.

In multicast network, a distributed mobile agent (MA) is deployed in each mobile host (MH) of the multicast tree. The main role of MA is to either broadcast the messages to all MHs in its cell or to a specific MH in its cell. However, MH can only transmit the messages to MA of the cell in which it is located.

The parameters used in this network model is taken as

- Mobile Host Identifier ($ID_{MH}$): Identity the mobile host
- First Level Node Identifier ($ID_{FN}$): Identity the immediate higher level nodes next to MH.
- Second Level Node Identifier ($ID_{FN}$): Identity the immediate higher level nodes next to first level nodes.
- Root Node Identifier ($ID_{RN}$): Identity the root node of the tree.
- Timer ($t$): Timer interval to detect whether the lower level nodes are in active or sleep mode.
- MA after identifying the neighbor list, it performs, monitoring the failures of multicast tree members.

## 2.1 Monitoring the Failures of Multicast Tree Members

The procedure for monitoring the failures of multicast tree members is mentioned below

1. Each Mobile Agent ($MA_i$) periodically transmits the 'Active' (AC) message only to its Mobile Host ($MH_i$.)

$$MA_i \xrightarrow{AC} MH_i$$

   This causes MA to be aware of active or failed MH among its immediate lower level nodes.
   *Note*: *Also MA decrements t for its immediate lower level nodes by one for every pre-defined time interval.*
2. Upon receiving AC message from $MA_i$, $MH_i$ sends active acknowledgement (AACK) message to $MA_i$.

$$MH_i \xrightarrow{AACK} MA_i$$

3. If $MA_i$ does not receive any AACK message from $MH_i$ until timer expiry, it concludes that $MH_i$ is subjected to failure.
   This technique of monitoring failure nodes results in low-failure free overhead.
4. If $MH_i$ wishes to have a new mobile agent to perform effective monitoring of the immediate next level nodes, it generates the new mobile agent $MA_{new}$.
5. $MA_{new}$ initiates the monitoring task and informs its location to all the nodes existing in between the $MH_i$ and itself.
6. When $MH_i$ decides that it is not efficient to guarantee the required monitoring performance, it forcefully shifts its $MA_i$ to other trusted $MH_j$, where $MA_i$ begins its monitoring task. If the MH detects that its immediate level nodes have failed, it initiates failure recovery technique

## 2.2 Failure Recovery Technique

If the MH detects that its immediate level nodes have failed, it initiates failure recovery technique. The recovery is based on the availability and capability of nodes. It involves following three scenarios:

**Scenario 1**
When MH identifies that its immediate low level nodes have failed, then

- Finds a new node as alternate node $N_{new}$
- Generates the new MA ($MA_{new}$)
- Transfers the $MA_{new}$ to $N_{new}$
- $MA_{new}$ performs the similar task as it was performing earlier.

**Scenario 2**
When MH identifies that its immediate lower level node has failed and there is availability of node for replacing the failed one, then

- It verifies its lower level nodes whether there is any suitable node for replacement.
- If node exists, then it is allowed to take over the role of failed MH
- The new node informs its location and its replacement to all the nodes in the path between the MH and failed node.

**Scenario 3**
When there is no new node or lower level nodes available for failed node replacement, then

- MH recommends the immediate higher level node to take the role of failed node.
- The new node informs its location and its replacement to all the nodes in the path.

## 2.3 Selecting the Caching Policy for Content Distribution Based on Game Theory Model

The caching policy is selected for content distribution based on game theory model which is explained below: When the mobile nodes receive the data from the Internet, they prefetch it, store, and share the content [8]. Also there is a possibility that the nodes may be subjected to dynamic content more than once.

Let $Q$ be the set of all websites that node have possibility to download
Let $CP$ be the caching policy.
$CP$: $\{n_i, n_j, \sigma, \tau\}$
$R$ represents the rule that selects four-tuple $\{n_i, n_j, \sigma, \tau\}$ to transmit at time slot $t$.
where $\{\sigma, \tau\}$ = time slot

$$\{n_i, n_j\} = \text{neighboring nodes.}$$

The steps involved in this approach are as follows:

1. Each node $n_i$ maintains the public cache where it stores the most downloaded $x_i$ websites, where
   $x_i \leq |Q|$.
2. The nodes make the contents of the cache to be available for other encountered nodes.
   i.e. If $n_i$ visits $n_j$, any website stored in $n_i$ will be made available for $n_j$ and vice versa.

$$\forall q \in Q, \text{ let } c_{ni,q} = \begin{cases} 1, \text{ if } n_i \text{ stores } q \\ 0, \text{ otherwise} \end{cases}$$

   Here, 0–1 variable indicates that if the node $n_i \in N$ stores websites $q$.
3. The caching policy (CP) of $n_i$ can be selected using the utility function ($UF_{ni}$) at time $t$.

$$UF_{ni}(CP, t) = \begin{cases} n_i(q(t)), \text{ if } n_i \text{stores } q \\ 0, \text{ otherwise} \end{cases}$$

4. For all $q \in Q$, $v_{i,q} = \sum_{n \in T_i} c_{ni,q} / |T_i|$ be the fraction of nodes in tree $T_i$ storing website $q$
   This reveals that $v_{i,q}$ is $v$'s replication ratio in tree $T_i$.

$$v_i = [v_i, q] \; q \in Q, \; i \in 1, \ldots, S$$

5. If the request is raised by the node for a website during time slot $(1-\sigma)$ {$0 \leq \sigma \leq 1$ of time slot}, a feasibility condition is defined as follows:

$$0 < (1-\sigma)\,(n_i q(t))$$

   The strict inequality assures that the request by the node is satisfied.
   i.e. If $CP_{n,\,q} = 1$

   Then
   Request is satisfied
   Else
   Request is backlogged
   End if

   Here $CP_{n,q} = 1$ reveals that the node stores the requested website in its public cache

The above condition will be subsequently satisfied during following scenarios:

- When $n_i$ meets $n_j$, such that $c_{nj,q} = 1$
- $n_i$ accesses the infrastructure

6. The optimal value of $\tau$ is estimated using following equation

$$\tau^* = \frac{(1-\sigma)CP^{nj}_{ninj}}{\sigma CP^{ni}_{nj}}$$

If $\tau < \frac{(1-\sigma)CP^{nj}_{ninj}}{\sigma CP^{ni}_{nj}}$
Then
$y$ does not have sufficient cache to store the website.
End if
If $\tau > \frac{(1-\sigma)CP^{nj}_{ninj}}{\sigma CP^{ni}_{nj}}$
Then
It is concluded that there is sufficient cache available.
End if

Thus the notation $R = \{n_i, n_j, \sigma\}$ is used for decision of caching policy.

7. In order to maximize the total expected utility of nodes while selecting caching policy, we perform the following

$$O(R,t) = \sum_{ni}^{N}\sum_{nj}^{N} UF^{Ni}_{ninj}(t) + UF^{Nj}_{ninj}(t)$$

**Note**:

- A request from $q$ can be satisfied when any copy of the website is retrieved.
- The above process becomes realistic during following scenarios:
  - When the websites are updated infrequently.
  - Content is diffused quickly among users that cache it.

## 3 Simulation Results

### *3.1 Simulation Parameters*

Authors used NS2 [9] to simulate our proposed Distributed Caching and Fault-tolerant Communication (DCFC) method. Authors used the IEEE 802.11 for Mobile Distributed Network as the MAC layer protocol. It has the functionality to notify the network layer about link breakage. In simulation, the packet sending rate

**Table 1** Simulation parameters

| No. of mobile nodes | 4, 6, 8, 10 and 12 |
|---|---|
| Area | 1250 × 1250 |
| MAC | 802.11 |
| Simulation time | 50 s |
| Traffic source | CBR |
| Rate | 150 Kb |
| Propagation | TwoRayGround |
| Antenna | OmniAntenna |

150 Kb. The area size is 1250 m × 1250 m square region for 50 s simulation time. The simulated traffic is constant bit rate (CBR) and exponential (Exp).

Our simulation settings and parameters are summarized in Table 1

## *3.2 Performance Metrics*

Authors evaluate performance of the new protocol mainly according to the following parameters:

**Average Packet Delivery Ratio** It is the ratio of the number of packets received successfully and the total Number of packets transmitted

**Average end-to-end delay** The end-to-end-delay is averaged over all surviving data packets from the Sources to the destinations

**Throughput** The throughput is the amount of data that can be sent from the sources to the destination

**Packet Drop** It is the number of packets dropped during the data transmission.

## *3.3 Results and Analysis*

### 3.3.1 Based on Mobile Nodes

In first experiment by varying the number of mobile nodes as 4, 6, 8, 10, and 12, the various factors are analysed and the simulation results are given below (Fig. 1).

Figures 2, 3, 4 and 5 show the results of delay, delivery ratio, packet drop, and throughput by varying the mobile nodes from 4 to 12 for the CBR traffic in DCFC and RMP protocols. When comparing the performance of the two protocols, we infer that DCFC outperforms RMP by 29 % in terms of delay, 42 % in terms of delivery ratio, 24 % in terms of packet drop, and 47 % in terms of throughput.

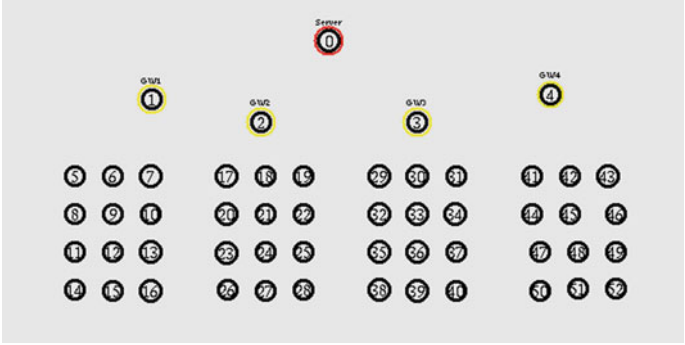



**Fig. 1** Simulation topology

**Fig. 2** Nodes versus delay

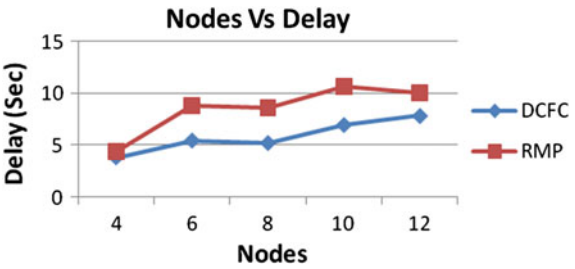



**Fig. 3** Nodes versus delivery ratio

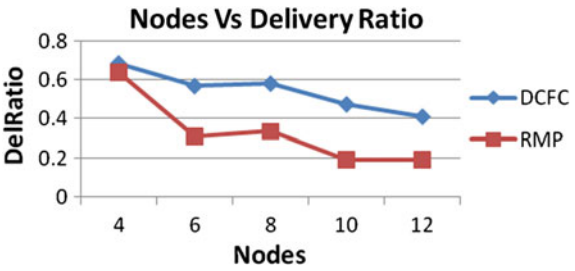
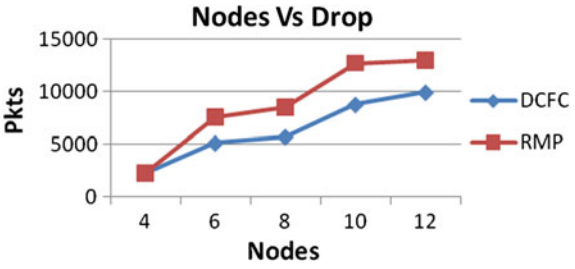



**Fig. 4** Nodes versus drop



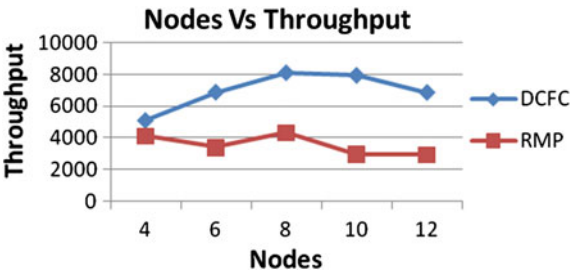



**Fig. 5** Nodes versus throughput

### 3.3.2 Based on Cache Size

In second experiment by varying the cache size as 50, 100, 150, 200, and 250, the various factors are analysed and the simulation results are given below.

Figures 6, 7, 8, and 9 show the results of delay, delivery ratio, packet drop, and throughput by varying the Cache Size from 50 to 250 for the CBR traffic in DCFC and RMP protocols. When comparing the performance of the two protocols, we infer that DCFC outperforms RMP by 18 % in terms of delay, 50 % in terms of delivery ratio, 16 % in terms of packet drop, and 53 % in terms of throughput.

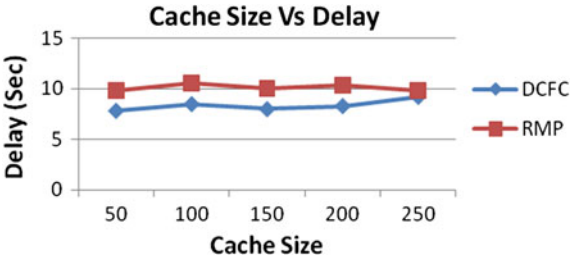



**Fig. 6** Cache size versus delay

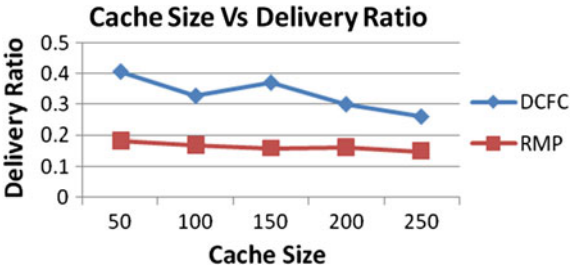



**Fig. 7** Cache size versus delivery ratio

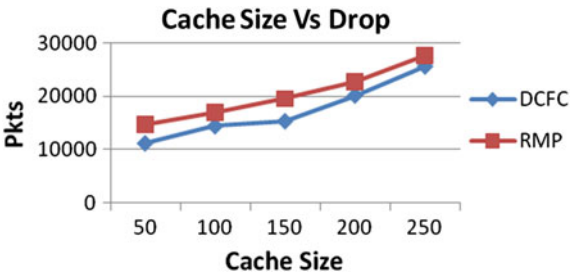



**Fig. 8** Cache size versus drop

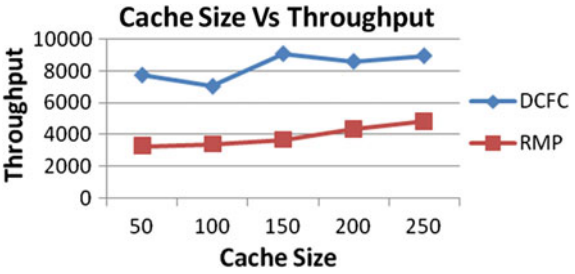



**Fig. 9** Cache size versus throughput

## 4 Conclusions

In this paper, authors have proposed a distributed caching and fault tolerant communication in mobile distributed networks. In this technique, a distributed mobile agent is deployed in each mobile host of the multicast tree. It initially monitors the failures occurring in multicast tree members. If the mobile host detects that its immediate level nodes have failed, it initiates failure recovery technique. Then the content distribution is performed by selecting the caching policy based on game theory. By comparing the protocols DCFC and RMP, the simulation results show that DCFC is better in performance with respect to delay, delivery ratio, throughput, and drop based on both the cache size and number of mobile nodes. Hence the proposed technique DCFC enhances the reliability and reduces the communication failures.

## References

1. D. BhuvanaSuganthi, "High Performance Mobile Computing Nodes in Distributed Networks", Volume 4, Issue 3, March 2014, http://www.ijarcsse.com.
2. Carlos Bobed, Sergio Ilarri, and Eduardo Mena, "Distributed Mobile Computing Development of Distributed Applications Using Mobile Agents", *PDPTA*. 2010.
3. WaseemAhmed and Yong Wei Wu, "A survey on reliability in distributed systems", Journal of Computer and System Sciences 79 (2013), Elsevier Inc. Pages 1243–1255.

4. Jinho Ahn, “Fault-tolerant Mobile Agent-based Monitoring Mechanism for Highly Dynamic Distributed Networks”, IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 3, May 2010.
5. Stratis Ioannidis, Laurent Massoulie and Augustin Chaintreau, “Distributed Caching over Heterogeneous Mobile Networks”, ACM SIGMETRICS Performance Evaluation Review. Vol. 38. No. 1. ACM, 2010.
6. Chien-Lung Hsu and Yu-Li Lin, “Improved migration for mobile computing in distributed networks”, Computer Standards and Interfaces, Volume 36. Issue 3, March 2014: Pages 577–584.
7. Giuseppe Anastasi, Alberto Bartoli and Francesco Spadoni, “A Reliable Multicast Protocol for Distributed Mobile Systems: Design and Evaluation”, Parallel and Distributed Systems, IEEE Transactions, 2001. Pages 1009–1022.
8. Huayong Wang and Li-ShiuanPeh, “MobiStreams: A Reliable Distributed Stream Processing System for Mobile Devices”, Parallel and Distributed Processing Symposium, IEEE 2014. Pages 51–60.
9. Network Simulator: http://www.isi.edu/nsnam/ns.