

# Enhanced Service Discovery Protocol for MANET by Effective Cache Management

Cynthia Jayapal<sup>1</sup> · Senthil Jayavel<sup>1</sup> · V. P. Sumathi<sup>1</sup>

Published online: 1 June 2018

© Springer Science+Business Media, LLC, part of Springer Nature 2018

**Abstract** With the increasing pervasiveness of the mobile nodes, service discovery has become inevitable. In the resource constrained mobile nodes, caching is being used to reduce the access cost. The enhanced service discovery protocol presented in this paper is a combination of cache placement, cache discovery, cache consistency and replacement algorithms. The overheard service information is cached by the coordinator nodes of each geographical area to reduce the discovery delay. A light weight cache discovery algorithm that uses a minimum spanning tree is proposed. The cache consistency is maintained in a soft state. Cache replacement is done based on a weight value that is calculated using the recency, frequency and cost factors. An  $M/G/1/\infty$  non-preemptive queuing model is proposed to analyse the cache replacement performance. The extensive simulation results performed to demonstrate the effectiveness of the proposed enhanced service discovery protocol in terms of reduction in the service discovery delay and number of cache replacements. The replacement scheme proposed has increased the cache hit ratio.

**Keywords** Service discovery · Cache placement · Cache discovery · Cache replacement · Ad hoc network · Queuing model

## 1 Introduction

Mobile ad hoc networks (MANET) are networks with mobile nodes that are characterized by mobility and energy limitations besides using multi hop communication. A service is any resource that is provided by one node and is utilized by other nodes. The role of a

---

✉ Cynthia Jayapal  
cynthia.j.it@kct.ac.in

<sup>1</sup> Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore, Tamil Nadu, India

service discovery (SD) protocol is to identify a resource of interest from a provider. In a directory based SD architecture, the service providers (SP) register their services with the directory nodes and the service information is provided to the requestors through these directories only. A directory may be maintained either in a centralized or distributed manner. The directory node acts as a coordinator that aids the service discovery process.

Caching is used to reduce the access cost and delay to overcome the resource constraints of an ad hoc network [1, 2]. The broadcast nature of wireless transmission enables all nodes within the transmission range to overhear and cache the transmitted information even though it is not the specified target. In this work, it is proposed to enhance the performance of the service discovery protocol in MANETs by effective cache management. The main issues involved in caching service information are cache placement, cache discovery, cache consistency and cache replacement. Cache placement in service discovery is defined as the location where the service provider information is cached. Cache discovery is discovering the location of a cache. Cache consistency is updating the provider information stored at all the cached locations. Cache replacement is replacing the existing service record when the service cache is full.

In the existing works, the location of caching is based on the access cost of the item to be cached or is decided cooperatively by all the caching nodes [3–5]. In these schemes, the overhead involved in deciding and sharing the location of the cached nodes is very high. Based on the above observations, a novel caching algorithm in which a coordinator node, caches the overheard service provider information is proposed. This coordinator node sends a reply to the service request received by it, instead of forwarding it to a distant coordinator.

Cache discovery is usually made either in a proactive, reactive or by using a combined query approach [5]. In a proactive discovery mechanism, the details of the caching nodes are maintained, and a request is forwarded to that node. This mechanism is not suitable for an ad hoc network owing to the transient nature of the mobile nodes. In reactive mechanisms, the requests are forwarded to the destination using traditional routing methods. A light weight cache discovery mechanism that uses an overlay minimum spanning tree (MST) to find the cached location is proposed in this work. The nodes of the MST are the zone coordinators where the service information is cached.

The cache replacement schemes usually based on policies such as frequency or recency of usage of a data item, and the size of the data item to be cached [6, 7]. The cache replacement is done based on two policies in the proposed work. The first one is deleting the service entries whose life time is less than the minimum service period and the second one is removing the service record with minimum weight. The proposed cache replacement scheme is modelled as an  $M/G/1/\infty$  non-pre-emptive priority queuing system.

The rest of the paper is organized as follows. In Sect. 2, the related works are discussed. Section 3, describes the service discovery mechanism using cache. Section 4, presents the proposed caching algorithm. Section 5 proposes a light weight cache discovery mechanism. In Sect. 6, two cache replacement schemes are presented, and its performance is analysed using  $M/G/1/\infty$  non-pre-emptive priority queuing system. Section 7 describes the simulation environment and presents the evaluation results. Finally, Sect. 8 presents the conclusion part.

## 2 Related Work

This section provides an overview of the techniques used in the literature for cache placement, discovery and replacement. Quite a lot of work has been done for cache placement, discovery and consistency in ad hoc networks [5]. The decision of where to cache an item is based on certain factors such as the terminal profile of a node or the item to be cached [1]. In cooperative caching schemes the location of caching and data access is decided in a cooperative manner so as to cache the data only in certain nodes based on its distance from the source. Cooperative caching techniques described in [8–10], improve the data access performance and availability in mobile ad hoc networks. However, variable data sizes, frequent data updates, limited client resources, insufficient wireless bandwidth and client's mobility make cache management a challenge [11]. Khawaga et al. [12], proposes administrative cluster-based cooperative caching that uses utmost two copies of the cached item in each cluster with a cluster manager and a cluster backup. Zam and Movahedinia [13], uses cluster based cache discovery mechanism that uses two cache tables known as Pre-fetched and Non-prefetched list in order to increase the cache hit ratio and cluster hit ratio. Joy and Jacob [14], describes push pull based cache discovery techniques and their respective advantages. Saleh [15], uses an adaptive cooperative caching strategy that forms non overlapping clusters using routing information.

Group based service discovery protocol [16], caches both service request and advertisement. Forward Node Minimization enhanced Group-based Service Discovery Protocol minimizes the number of next hop nodes when forwarding the request packets by making use of the service advertisement information in the cache [17]. A service cache can be maintained at each node [18] or in the backbone nodes [19] or selective nodes based on cache space available or popularity of data item cached [1]. In the caching schemes that maintain the cache in all the nodes, the overhead at the time of service advertisement is increased and leads to cache consistency problem.

Cache maintenance can be done either in a hard state, where any change in information must be explicitly intimated to all the nodes caching the information. Tian et al. [20], Cao [21] and Madhukar et al. [22] send invalidation reports to overcome the cache consistency problem either periodically or whenever there is a change in the cached information. In an ad hoc network cache maintenance in a soft state, where a cached entry is deleted after a time period, is preferred to reduce the communication overhead and to handle mobility.

The cache replacement is done based on certain factors such as LRU, data size, transfer time, data invalidation rate and access interest [23]. Cache replacement based on least recently used (LRU) policy replaces the entries stored in the cache based on the recency of references. It replaces the block in the cache which has not been used for the longest period of time. This is found by using the age bits. The least frequently used (LFU) policy replaces the entries based on the frequency of references. It replaces the block that is least frequently used. It is done by maintaining a reference count. Literature also uses a combination policy called LRFU that inherits the benefits of both of the above policies by leaning towards the LRU or the LFU by usage of a weighing parameter [6, 7].

### 3 Service Discovery with Cache Maintenance

In a MANET hierarchical decomposition the geographic area is done to divide an area into zones and to facilitate communication amongst the mobile nodes. The zone coordinators (ZC) maintain the information predominantly within its zone, and the collective information of all the zones are maintained by a single area coordinator (AC). To facilitate the SD process, these coordinators keep the details about the service, location and membership information of the providers and the requestors.

The crucial issue in directory based architecture is the selection of a coordinator node. These coordinators are selected adaptively to ensure that they are not drained, and also additional backup nodes may provide support in case of failure of the coordinators. To investigate the effect of the proposed cache management schemes on the SD protocol in MANET, the coordinator election, service discovery and delivery are done as described in [24]. An SP advertises its services to the ZC and AC. When a service requestor (SR) makes a request to its ZC, it is forwarded to the provider if the location of the provider is known; otherwise, the request is sent to the AC. Any ZC en-route to the AC may forward the request to the provider if the SP location is cached.

Figure 1, shows the sequence diagram of the service discovery process with and without using the cache. When a service request is sent to a ZC, the ZC checks its ZCTABLE for the SP information. If it is not found, the ZC forwards the request to the AC, through the intermediate ZCs. Any intermediate ZCs, receiving the service request checks the

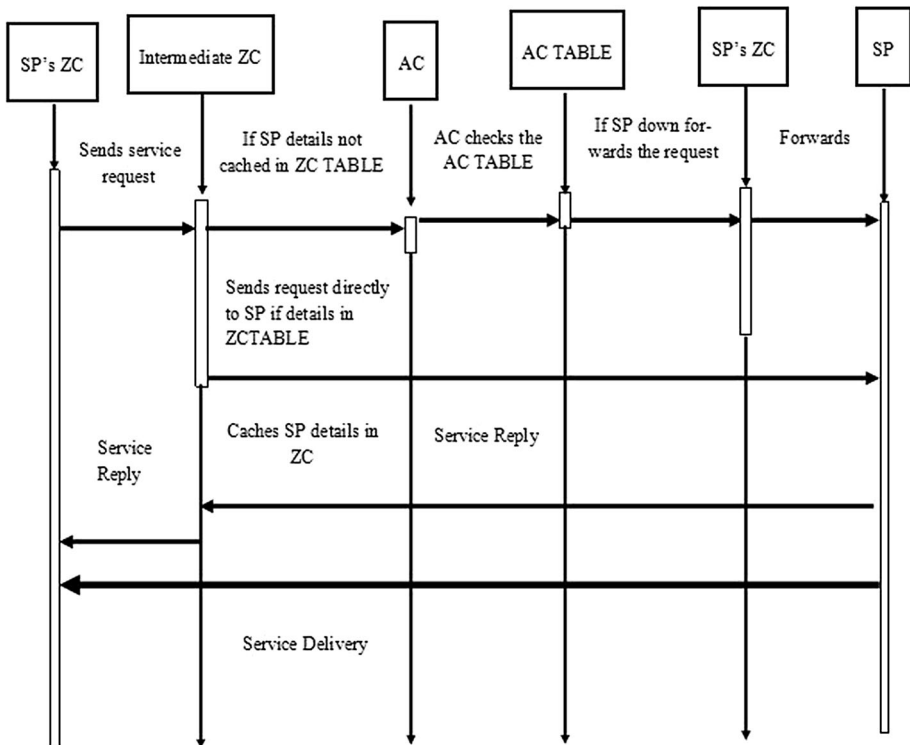


Fig. 1 Service discovery process with and without cache

ZCTABLE for the cached service record. If it is found, the request is sent directly to the SP to reduce the discovery delay. If the SP information is not found, the request is forwarded to the adjacent ZC in the path leading to the AC. This path is computed using the minimum spanning tree algorithm described in Sect. 5. The AC finally forwards the request to the SP. The SP sends the service reply to the SR's ZC. The intermediate ZCs that over hears this reply updates their service cache. The SP delivers the service to all the SRs, through their respective ZCs periodically.

---

Algorithm: service discovery using caching

---

The notations used in the algorithm are as follows

$S_{id}$ : A service for which a request has been sent or a reply has been received

$C_{id}$ : A zone or area coordinator that receives the request or reply

$P_{id}$ : A service provider providing the service

$R_{id}$ : A service requestor requesting the service

$S_{rq}$ : The service request

AC: Area Coordinator

ZC: Zone coordinator

The  $x$ -best path between the service requestor's zone coordinator (ZC) and the area coordinator (AC) that is found using the Prim's algorithm

$x^*$  -previously obtained the best path between SR's ZC and AC

$C(x)$  -Cost function that gives the total cost of the path  $x$

$C(x^*)$ -Cost function that gives the total cost of the path  $x^*$

$k$  -Frequency count

Repeat until  $P_{id}$  is found.

When a service request  $S_{rq}$  from a requestor  $R_{id}$  is heard by a coordinator  $C_{id}$ :

If ( $S_{id}$  in cache) then

Send  $S_{rq}$  to  $P_{id}$  and increment  $k$

Else

Find  $x$  using  $\text{minspmtree}(\text{SR's ZC, AC})$

If  $C(x) < C(x^*)$  then

Set  $x^*$  as  $x$

Send  $S_{rq}$  to  $C_{id}$  that is higher up in the hierarchy in the path  $x$  leading to AC.

End if

---

## 4 Caching the Service Information

The SPs in a zone wishing to provide any service, register their service with the respective ZC by sending the associated service parameters. The ZC upon receiving this information updates the ZC TABLE in Table 1. The ZCs of all the zones, advertise the ZCTABLE

**Table 1** ZCTABLE

SR's ID	SP's ID	SR location	SP location	Service parameters
---------	---------	-------------	-------------	--------------------

contents, to the AC periodically and the AC, in turn, updates its ACTABLE. The coordinator nodes use the ZCTABLE and ACTABLE in Table 2 as a service cache.

The service parameters maintained in the cache are service ID, the life time of the service, sequence number, frequency count, and combined recency, frequency, cost value (CRFC). Each service reply has a unique sequence number and is used by the coordinator nodes to identify the reply messages uniquely. The frequency count indicates the number of times the cache has serviced a request. The CRFC value is the weight value attached to each of the service records. The significance of this CRFC value is described later in Sect. 6.

When a service reply is heard by a coordinator, and no entry for that service is found in the cache (ZCTABLE), the service record is cached. If an entry for the service is already available, then the entry for the provider that is nearest to the ZC is retained. If the entry for the same service and provider is available, the entry from the latest reply is stored by checking the sequence number of the reply message. Cache replacement is needed if there is no free space in the cache to store the service record. Since the adaptive leader election proposed in [24] is used for electing the coordinator nodes, the cached information is transferred during handoffs to the new coordinator nodes. The algorithm for service caching is given below.

---

#### Algorithm: Service caching

---

The notations used in the algorithm are as follows

$S_{rp}$ : The service reply

$S_n$ : The sequence number of the service reply message

$D(C_{id}, P_{id})$ : The distance between a coordinator  $C_{id}$  and provider  $P_{id}$

$n$ : reference to cache size which is initialized as  $n=0$  and reaches till  $C_{max}$

$C_{max}$ : Maximum Cache size

$l$ : Length of a service record

When a  $S_{rp}$  is received or over heard by  $C_{id}$ :

If ( $S_{id}$  is in cache) then

If ( $S_{rp}$  is from a new  $P_{id}$ ) then

If ( $(D(C_{id}, P_{id-new}) > D(C_{id}, P_{id-old}))$ ) then

    Replace the cache entry of existing  $P_{id}$  with that from new  $P_{id}$ . Return.

/\*Store the service parameters of the nearest provider\*/

Else If ( $(S_{rp}$  is from existing  $P_{id}$ ) and ( $S_n$  of received  $S_{id} > S_n$  of  $S_{id}$  in cache)) then

    Replace old entry in the cache. Return.

/\*Store the latest reply with higher sequence number\*/

Else If ( $n < C_{max}$ )

    Store the  $S_{id}$  with its associated service parameters in the cache.

/\*If  $S_{id}$  not in cache\*/

$n = n + 1$  /\* Increment n by size of the service record\*/

Else

    Replace ( )

    /\* Replacement occurs since maximum cache size is reached\*/

    Store  $S_{id}$  with the associated service parameters in the cache

End if

---

**Table 2** ACTABLE

ZC's ID	ZC's location	ZC's zone ID	Service parameters
---------	---------------	--------------	--------------------

## 5 Light Weight Service Discovery

Service cache discovery is discovering the location of the cache where the service provider information is stored. Traditional proactive approaches for discovering the location of the cache, by storing the information and location about the caching nodes is not suitable for MANETs, because of the frequent change in the position of the mobile nodes. When an SR is received by a ZC, it forwards the request to the AC. Therefore the ZC must know the path through which a service request must be forwarded. A minimum spanning tree (MST) algorithm is used to find the cache path if the provider information is not available with a ZC that has received the service request.

In the proposed light weight cache discovery mechanism, we use an overlay MST that connects all the ZCs and the AC. There is no additional overhead in tree construction and maintenance because the MST that is used for SD is also used for cache discovery. An overlay tree is an MST that connects only the coordinator nodes, and the forwarding nodes are computed only at the time of transmission and are hence more suitable for an ad hoc network. Prim's algorithm is used for the construction of MST. The advantages of using Prim's algorithm are given in [25]. The minimum spanning tree algorithm that is used to find the location of the cache is described below.

---

### Algorithm: Cache Discovery

---

minspree( SR'sZC,AC )

/\*The algorithm starts with a tree consisting of a single vertex(SR's ZC), and adds one edge at a time until it spans all vertices (all the ZC's and AC)to construct an overlay spanning tree based on Prim's Algorithm \*/

- Input: A non-empty weighted graph with the vertices  $N_v = \{AC, ZC_1, ZC_2 \dots ZC_n\}$  where  $n$  is the number of geographic zones, the Intermediate edges  $I_e$ : the set of edges between every pair of nodes from the set  $N_v$  and the the set  $C_p =$  distance between node  $i$  and  $j$  (Dij).
  - Initialize: Path = {AC},  $I_e = \{\}$ ,  $C_p = \{\}$ ,  $T_c = 0$ .
  - Repeat until Path =  $N_v$ : (i.e., until all the ZC's and AC get connected in the tree)
    - Choose an Intermediate edge( $I_e$ ) {curr\_node,next\_node} such that curr\_node is in Path{} and next\_node is not in the Path{}
    - Add next\_node to Path{}, and {curr\_node, next\_node} to  $I_e$
    - Set next\_node as curr\_node
    - Compute the cost of the path as  $T_c = T_c + D_{curr\_node,next\_node}$
  - Output: Path{ $n_1 \dots n_z$ },  $I_e$  describe a minimal spanning tree, and  $T_c$  specifies the total cost of the new path.
-

## 6 Cache Consistency and Replacement

While caching dynamic information, the data becomes obsolete after a specified time. The service information cached in the coordinator has to be updated periodically.

### 6.1 Cache Replacement Algorithm

During advertisement of the service information, the provider specifies the time period for which a service is available. The coordinators calculate the minimum service time that is required for sending a request to the provider and receiving the service from the provider based on the distance from the coordinator to the provider. When a ZC hears a new service request, it stores it in the ZCTABLE. If there is a lack of space in the cache, all the entries whose life times are less than the minimum service period are deleted. If more free space is still required, the entry with the minimum weight value is removed. This CRFC value is calculated by taking into consideration the frequency, recency and cost factors of a cached service record. The size of the data item is not considered since all the service records are of the same size. The algorithm for cache replacement is given below.

---

Algorithm: Cache Replacement

---

$T_{min}$ : Minimum service time is taken for a request to be serviced  
 Replace ( $\cdot$ )  
 Repeat for all service records in the cache  
     If (Life time of  $S_{id} < T_{min}$ ) then  
         Remove the entry from cache  
          $n=n-1$   
     else  
         Update crfc value for all the entries  
     endif  
 If ( $n= C_{max}$ )  
     Remove the entry with minimum crfc value  
     Update crfc value for all the entries  
 endif

---

### 6.2 Computation of CRFC value

The CRFC value for each of the service record is calculated as shown in Eq. (1).

$$W(s) = R(0) + F(s) * A(s) \tag{1}$$

The Eq. (1) can be expanded based on Kim, C.S. 2001 and Li, Z et al. 2012 as follows,

$$W_{t_i}(s) = R(0) + F(t_s - t'_s) * W_{t'_i}(s) * A(s) \tag{2}$$

where  $R(0)$ , Weight value of recently referenced service record;  $F(s)$ , Frequency of reference to the service record;  $A(s)$ , Access cost for fetching a service record  $s$  from the AC;  $W_{t_i}(s)$ , Weight value of a service record  $s$  at the time of  $i$ th reference;  $W_{t'_i}(s)$ , Weight value of a service record  $s$  at time of  $i - 1$ th reference;  $t_s$ , Time stamp for the current reference to a service record  $s$ ;  $t'_s$ , Time stamp for the previous reference to a service record  $s$ .



The combined least frequently recently policy associates a value with each block in the cache. This value gives the likelihood that the block will be referred shortly. Each reference to a block in the past contributes to this value and a reference's contribution is determined by a weighing function  $F(x) = e - \gamma x$  where  $x$  is the time span from the reference in the past to the current time. When  $\gamma = 0$ ,  $F(x) = 0$  and the function behaves as LFU policy. When  $\gamma = 1$ ,  $F(x) = e - x$  and the function behaves as LRU policy. It is found that the performance is optimum when  $\gamma = 0.5$  [6]. The cost value of an object at the time of  $k$ th reference can be computed from the time of  $(k - 1)$ th reference and the cost value at that time. The block with the minimum value is replaced [6, 7]. In the proposed scheme, the cost factor is added to the recency and frequency factors to compute the CRFC value. This weight value quantifies the likelihood that a service record stored in the cache will be referred shortly.

### 6.3 Modelling Cache Replacement as M/G/1/ $\infty$ Non-Pre-Emptive Priority Queuing Model

To explore the effectiveness of the cache replacement strategy, it is modelled as an M/G/1/ $\infty$  with non-pre-emptive priority queuing model.

M/G/1 queuing system is a single server system in which the arrivals are Poisson with rate  $\lambda > 0$  and the service time are distributed and are independent of the arrival time. The mean service time is given by  $S = 1/\mu$ . The number of servers is 1, and the size of the server capacity is  $\infty$ . Usually, the customers are served according to FCFS service discipline. In applications where preferential treatment has to be given, priority queues are considered. In priority queues, there are at least two priority classes. If the priority is lower, the class number is greater. The customers of the same priority are served in FCFS order. The priority policy may be pre-emptive or non-pre-emptive. In pre-emptive policy, a customer that is being served is pre-empted by the newly arrived customer, bringing the customer that is currently served to the beginning of the priority queue and is the next customer to be served. In non-pre-emptive policy, the customer who is serviced is not interrupted by a higher priority customer that has arrived.

The overhearing of an SA by the coordinator node is considered as the Poisson arrival of an SA. The cache replacement is considered as service. The time taken for each service is distributed because the time taken to remove an old entry when its service time expires and the time taken to calculate the minimum weight value and remove it, is independent of the arrival of an SA.

Since a preferential treatment is given for cache replacement based on its class, the system is modelled as a priority queue system; with non-pre-emptive priority policy as a new SA that arrives is never a candidate for replacement. The priorities for cache replacement are classified as follows,

Class 1: SA whose service lifetime has expired; has the highest priority to be removed from the cache.

Class 2: SA whose weight value is the least; has the second highest priority to be replaced from the cache.

As each coordinator maintains only one cache, it is modelled as a single server system as shown in Fig. 2. In an M/G/1 system  $\lambda_{\text{eff}} = \lambda$ , since there are no blocked customers and there are no SA that is not stored in the cache because of the finite cache size. Hence a  $\infty$  queue length assumption is made. An SA has to wait for a residual service time before it enters the cache.

Modelling of Cache Replacements using queuing theory approach:

Notations:

- $\mu_i$  Replacement rate of a SA of class  $i$  from the cache
- $E[\omega_i]$  Mean waiting time for a SA of class  $i$  before it is replaced
- $E[R]$  Mean residual service time of a SA
- $E[X_i]$  Mean replacement time of a SA of Class  $i = 1/\mu_i$
- $E[X_{i2}]$  Second moment of the replacement time
- $\rho_i$  Replacement intensity of a SA of class  $I = \lambda_i/\mu_i$

Mean Residual service time for an SA of class  $i$  is the remaining time taken for an advertisement to be removed from the cache when a new advertisement arrives at the cache. If no advertisement is currently being removed then  $E[R_i] = 0$ .

$$E[R] = \sum_{i=1}^n \frac{\lambda E[X_i^2]}{2} \tag{3}$$

where  $n$  is the total number of priority classes

Mean waiting time of an SA in the cache is the mean time for which an SA resides in the cache.

$$E[\omega_i] = \frac{E[R]}{(1 - \mu_i)(1 - \mu_i - 1)} \tag{4}$$

where  $\mu_i - \sum_{j=1}^i \rho_j$  We get,

$$E[\omega_i] = \frac{\sum_{i=1}^n \frac{\lambda E[X_i^2]}{2}}{(1 - \mu_i)(1 - \mu_i - 1)} \tag{5}$$

Mean Sojourn time of an SA of class  $i$  in the system, is the mean time for which an SA resides in the system.

$$E[T_i] = \frac{1}{\mu_i} + E[\omega_i] \tag{6}$$

We get,

$$E[T_i] = \frac{1}{\mu_i} + \frac{\sum_{i=1}^n \frac{\lambda E[X_i^2]}{2}}{(1 - \mu_i)(1 - \mu_i - 1)} \tag{7}$$

Applying Little’s formula to (3), the average number of SA of class  $i$  in the cache is

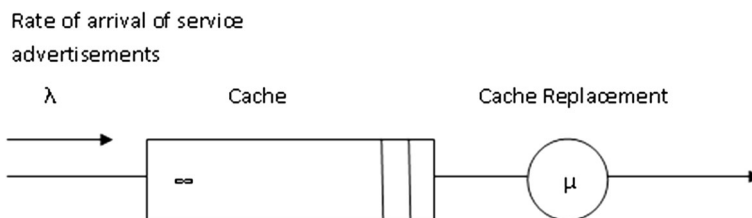


Fig. 2 M/G/1/∞ model for cache replacement

$$E[N_i] = \rho_i + \lambda_i E[\omega_i] \quad (8)$$

We get

$$E[N_i] = \rho_i + \lambda_i \frac{\sum_{i=1}^n \frac{\lambda E[X_i^2]}{2}}{(1 - \mu_i)(1 - \mu_i - 1)} \quad (9)$$

The probability of a service advertisement being in a cache of size  $z$

$$\frac{\frac{\rho^z}{z!}}{1 + \rho + \frac{\rho^2}{2!} + \dots + \frac{\rho^z}{z!}} \quad (10)$$

## 7 Performance Evaluation

### 7.1 Simulation Setup

The simulation was done using ns2 simulator [26]. The scenario used for ad hoc network simulation is given in Table 3. A result was gained by an averaging number of simulation runs with different seed values. The network size was varied from 30 to 50, and the number of service requests is increased proportionately.

The service distribution model is set randomly as follows. 25% of the mobile nodes act as service providers. A total number of services provided is 75% of the nodes. An SP may provide a maximum of 5 different types of services. For service and location update, and beacon packets the CBR packet size is set as 100 bytes and for service registration, advertisement, request, reply and update exchanges the CBR packet size is set as 512 bytes.

### 7.2 Simulation Results

#### 7.2.1 Effect of Cache on Service Discovery

To explore the effect of cache on service discovery, service discovery framework with and without using cache, for varying network size are simulated. Average discovery delay is the average time interval between a node issuing a service request and receiving the corresponding acknowledgement message. Figure 3, shows that the use of cache reduces the discovery delay by 41% because if any intermediate LC locates the SP information in the cache, it forwards the request to the provider instead of to the GC. Also, it is found that the discovery delay increases with the increase in the network size. When the network size increases, the number of service requests is increased proportionately in the simulation and hence the average discovery delay also increases.

#### 7.2.2 Effect of Cache Size on the Cache Performance

To explore the effect of the cache size on cache hit ratio, the simulation was run for 900 ms with varying pause times. A cache hit ratio is the ratio of the number of cache hits to the total number of issued request message. Figure 4, shows the effect of cache size on cache hit ratio. When the cache size is large, number of service records can be stored in the

cache, and hence the hit ratio improves as the cache size is increased. Pause time is the time for which the mobile nodes stay stationary before starting its movement in the random way point mobility model used in simulation. When the pause time is less mobility is more. When the cache size is large, more number of service records can be stored in the cache and hence the hit ratio improves as the cache size is increased. It is also found that when mobility is more the cache hit ratio reduces because of the change in service record information stored in the cache. The cache hit ratio is 43% greater when pause time is 40 s when compared to the hit ratio when pause time is 20 s.

To explore the effect of cache size on the service discovery performance, average discovery delay is determined for the entire simulation duration. It is found from Fig. 5 that when the cache size increases, discovery delay reduces because more number of service records can be stored in the cache. When a service request is received and the provider information is found in the cache the request is directly forwarded to the provider, instead of forwarding to the GC, which forwards the request to the provider. The provider then sends the corresponding acknowledgement message. It is also found that the discovery delay is more when the mobility is more because of the change in the service record information. The average discovery delay is 61% greater when pause time is 20 s when compared to the delay when pause time is 40 s.

To explore the effect of the cache size on the number of replacements, the simulation was run for 900 ms using the three replacement schemes. A service reply is cached, by the coordinator nodes and a service record stored in the cache is replaced when the cache is full. The cache replacement scheme proposed in our work is named as Combined Recency Frequency and Cost (CRFC). Figure 6, show that the average number of replacement using CRFC is 18% lesser than LFU and 12% lesser than LRU. This is because the weight value considered for replacement, takes into accounts the frequency, recency and access cost into account. It is found that the number of replacements is less when the cache size is more because more number of service records is stored in the cache.

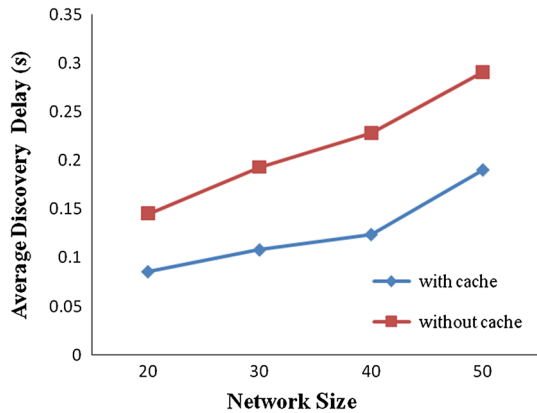
### 7.2.3 Effect of the Network Size on the Cache Performance

To explore the effect of network size on the number of replacements, the number of nodes is varied from 30 to 50. The cache size is kept as constant. From Fig. 7, it is found that the number of replacement increases with the increase in number of nodes. In the simulation, the number of service requests increases in proportion to the number of nodes and hence

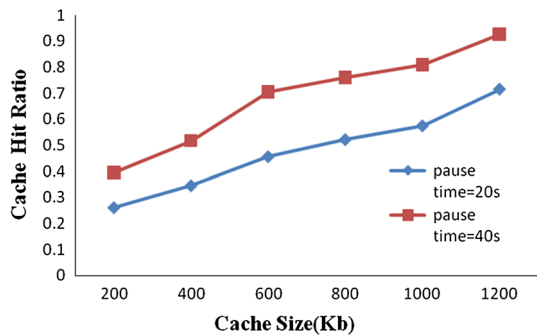
**Table 3** Scenario for simulation

Parameter	Values
Area	2400 m × 2400 m
Zone size	800 m
MAC protocol	IEEE 802.11
Mobility model	Random way point
Propagation	Two ray ground
Radio range	250 m
Traffic pattern	CBR
Energy model-initial energy	0.5 J
Transmitting power	0.6 W
Receiving power	0.3 W

**Fig. 3** Effect of network size on average discovery delay



**Fig. 4** Effect of cache size on cache hit ratio



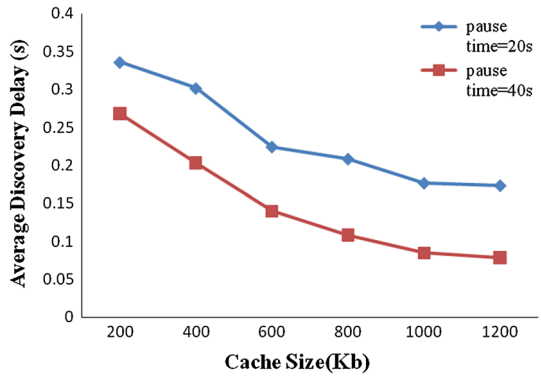
the number of replacements are more. The number of replacements using CRFC is 15% less than LRU and 28% less than LFU. This is because the weight factor considered for cache replacement ensures that the service records that have the likelihood of access in the near future is retained in the cache.

#### 7.2.4 Effect of Mobility on Average Discovery Delay

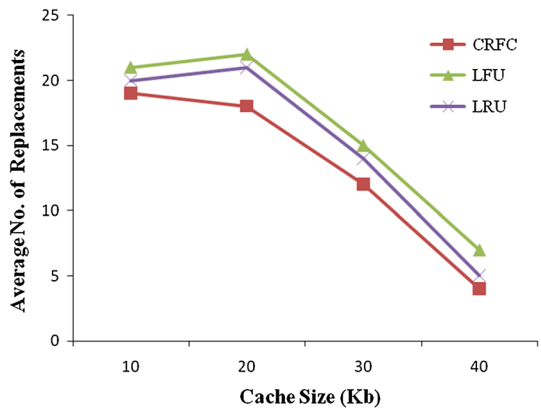
Figure 8, shows the effect of cache replacement schemes on the discovery delay. The simulation was run at different speed and delay was calculated using three cache replacement schemes namely LFU, LRU and CRFC. Discovery delay increases proportionately with speed. This is because, when the speed increases the SP location changes and the SP takes more time to update its service record stored in the cache. It is also found that the CRFC scheme is 20% lesser than LFU and 38% lesser than LRU.

From Fig. 9, it is found that the cache hit ratio is less when mobility is more. This is because of the change in the service parameter values stored in the cache and the change in the location of the SP. The cache hit ratio increases with increase in the cache size using CRFC. To explore the effect of speed on cache hit ratio, the speed is varied and the cache hit ratio is determined. The hit ratio of CRFC is 20% greater than LFU and 38% greater than LRU.

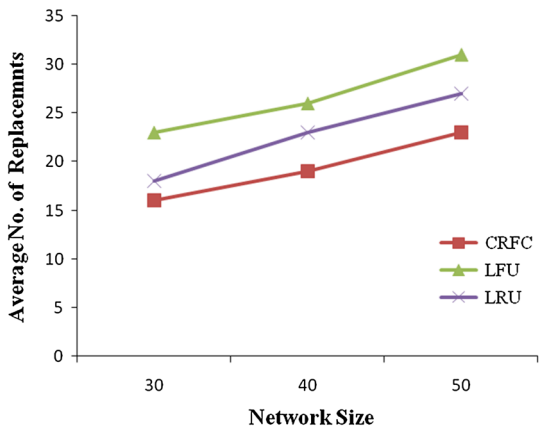
**Fig. 5** Effect of cache size on average discovery delay



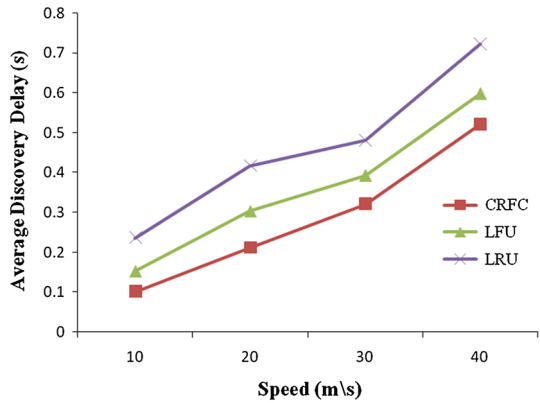
**Fig. 6** Effect of cache size on the average number of replacements



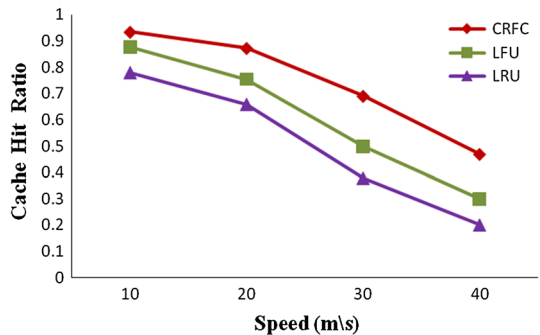
**Fig. 7** Effect of network size on the number of replacements



**Fig. 8** Effect of speed on the average discovery delay



**Fig. 9** Effect of speed on cache hit ratio



## 8 Conclusion

To evaluate the performance of our proposed algorithm, extensive simulations were done. The service discovery framework, with and without using cache was simulated. The caching algorithm proposed in this paper uses the over hearing property of a wireless node to cache the service reply from the providers at the coordinator nodes. It was found that the caching reduces the discovery delay considerably. The light weight MST algorithm used for cache discovery and the weight value based cache replacement scheme used for cache management has improved the performance of caching in terms of discovery delay; a cache hit ratio and the number of replacements.

## References

- Sailhan, F., & Issarny, V. (2003). Cooperative caching in ad hoc networks. In M. S. Chen, P. K. Chrysanthos, M. Sloman, & A. Zaslavsky (Eds.), *International conference on mobile data management* (pp. 13–28). Berlin: Springer.
- Lee, D., Choi, J., Kim, J. H., Noh, S. H., Min, S. L., Cho, Y., & Kim, C. S. (1996). LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies. *IEEE Transactions on Computers*, 50(12), 1352–1361.

3. Tang, B., Gupta, H., & Das, S. R. (2008). Benefit-based data caching in ad hoc networks. *IEEE Transactions on Mobile Computing*, 7(3), 289–304.
4. Yin, L., & Cao, G. (2004). Supporting cooperative caching in ad hoc networks. In *INFOCOM 2004. Twenty-third annual joint conference of the IEEE computer and communications societies* (Vol. 4, pp. 2537–2547). IEEE.
5. Wu, W., Cao, J., Yang, J., & Raynal, M. (2007). Design and performance evaluation of efficient consensus protocols for mobile ad hoc networks. *IEEE Transactions on Computers*, 56(8), 1055–1070.
6. Kim, C. S. (2001). LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies. *IEEE Transactions on Computers*, 50(12), 1352–1361.
7. Li, Z., Simon, G., & Gravey, A. (2012). Caching policies for in-network caching. In *2012 21st International conference on computer communications and networks (ICCCN)* (pp. 1–7). IEEE.
8. Sharma, K., & Raut, A. (2013). Survey on performance enhancement in mobile adhoc networks using data caching schemes. *Information Communication and Embedded Systems*, 2, 95–98.
9. Chuang, P., Chen, Y., & Chen, H. (2013). Efficient cooperative caching in mobile ad-hoc networks. In *Proceedings of the 2nd international conference on cloud-computing super-computing CCSC* (Vol. 2, pp. 104–108).
10. Jain, D., & Sharma, S. (2014). Basic needs for designing a good cooperative caching management technique. *International Journal of Computers and Applications*, 105(14), 22–25.
11. Chand, N., Joshi, R. C., & Misra, M. (2007). Cooperative caching in mobile ad hoc networks based on data utility. *Mobile Information Systems*, 3(1), 19–37.
12. Khawaga, S. E., Saleh, A. I., & Ali, H. A. (2016). An administrative cluster-based cooperative caching (ACCC) strategy for mobile ad hoc network. *Journal of Network and Computer Applications*, 69(3), 54–76.
13. Zam, A., & Movahedinia, N. (2013). Performance improvement of cache management in cluster based MANET. *International Journal of Computer Network and Information Security*, 10(1), 24–29.
14. Joy, P., & Jacob, K. (2012). Cache replacement policies for cooperative caching in mobile ad hoc networks. *International Journal of Computer Science Issues*, 9(3), 500–505.
15. Saleh, A. I. (2017). An adaptive cooperative caching strategy (ACCS) for mobile ad hoc networks. *Knowledge-Based Systems*, 120, 133–172.
16. Chakraborty, D., Joshi, A., Yesha, Y., & Finin, T. (2002). GSD: A novel group-based service discovery protocol for MANETS. In *4th International workshop on mobile and wireless communications network, 2002* (pp. 140–144). IEEE.
17. Gao, Z., Wang, L., Yang, M., & Wang, J. (2011). FNMGS DP: An optimized group-based service discovery protocol for MANETs. *Wireless Personal Communications*, 57(2), 137–162.
18. Xiang, X., Zhou, Z., & Wang, X. (2007). Self-adaptive on demand geographic routing protocols for mobile ad hoc networks. In *Proceedings of IEEE international conference on computer communications, Alaska* (pp. 2296–2300).
19. Xiang, X., Zhou, Z., & Wang, X. (2007). Robust and scalable geographic multicast protocol for mobile ad hoc networks. In *Proceedings of IEEE international conference on computer communications, Alaska* (pp. 2301–2305).
20. Tian, K. L., Cai, J., & Ooi, B. C. (2001). An evaluation of cache invalidation strategies in wireless environments. *IEEE Transactions on Parallel and Distributed Systems*, 12(8), 789–807.
21. Cao, G. (2003). A scalable low-latency cache invalidation strategy for mobile environments. *IEEE Transactions on Knowledge and Data Engineering*, 15(5), 1251–1265.
22. Madhukar, A., Özyer, T., & Alhajj, R. (2009). Dynamic cache invalidation scheme for wireless mobile environments. *Wireless Networks*, 15(6), 727–740.
23. Yin, L., & Cao, G. (2006). Supporting cooperative caching in ad hoc networks. *Proceedings of IEEE Transactions on Mobile Computing, USA*, 5(1), 77–89.
24. Jayapal, C., & Vembu, S. (2011). Adaptive service discovery protocol for mobile ad hoc networks. *European Journal of Scientific Research*, 49(1), 6–17.
25. Das, S. M., Pucha, H., & Hu, Y. C. (2008). Distributed hashing for scalable multicast in wireless ad hoc networks. *IEEE Transactions on Parallel and Distributed Systems*, 19(3), 347–362.
26. NS-2 Simulator. <http://www.insl.edu/nsname/ns>. Accessed December 10, 2010.

## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.





**Cynthia Jayapal** acquired her B.E. degree in Computer science and Engineering from Periyar Maniammai College of Technology, Bharathidasan University in the year 1994 and Master's degree in Computer Science and Engineering from Kumaraguru College of Technology, Bharathiar University Coimbatore, in the year 2001. She has obtained her Ph.D. in Information and Communication Engineering from Anna University. She has got over 20 years of experience. At present she is a Professor in the Department of Computer Science and Engineering, Kumaraguru College of Technology, Coimbatore, India. Her area of interests includes wireless communication and data communication in IP networks. She has published over twenty papers in International Conferences and Journals.



**Senthil Jayavel** acquired his B.E. degree in Computer science and Engineering from Kongu Engineering College, Bharathiar University in 2004 and Master of Computer science with Honors from Illinois Institute of Technology, Chicago, IL, USA in 2007. He has obtained his Ph.D. in Automated Computing and Software Engineering from Anna University, Chennai in the year 2014. He has over 11 years of teaching, research and industrial experience in renowned institutions in India and in USA. He has broad experience in automated, semantic computing, software engineering, Cloud Computing and Big Data Analytics in social physics. He has published over twenty papers in International Conferences and Journals.



**V. P. Sumathi** is presently working as Assistant Professor (SRG) at Kumaraguru College of Technology in the Department of Computer Science and Engineering, Coimbatore, Tamil Nadu, India. She has 13 years of teaching experience. She has around 14 publications in various journals and conferences. Her research interests include Semantic web, Data mining and Bigdata analytics.