IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 1, January 2011
ISSN (Online): 1694-0814
www.IJCSI.org

An Efficient Searching and an Optimized Cache Coherence handling Scheme on DSR Routing Protocol for MANETS

Mr. Rajneesh Kumar Gujral[1], Dr. Anil Kapil[2]

[1]Assoc. Professor, Computer Engineering Department, M. M. Engineering College, M. M. University, Ambala, Haryana, India-133207.

[2]Professor M. M. Institute of Computer Technology and Business Management, M. M. University, Ambala, India-133207.

Abstract

Mobile ad hoc networks (MANETS) are self-created and self organized by a collection of mobile nodes, interconnected by multi-hop wireless paths in a strictly peer to peer fashion. DSR (Dynamic Source Routing) is an on-demand routing protocol for wireless ad hoc networks that floods route requests when the route is needed. Route caches in intermediate mobile node on DSR are used to reduce flooding of route requests. But with the increase in network size, node mobility and local cache of every mobile node cached route quickly become stale or inefficient. In this paper, for efficient searching, we have proposed a generic searching algorithm on associative cache memory organization to faster searching single/multiple paths for destination if exist in intermediate mobile node cache with a complexity $O(n)$ (Where n is number of bits required to represent the searched field).The other major problem of DSR is that the route maintenance mechanism does not locally repair a broken link and Stale cache information could also result in inconsistencies during the route discovery /reconstruction phase. So to deal this, we have proposed an optimized cache coherence handling scheme for on -demand routing protocol (DSR).

Keywords: *DSR, Efficient Searching, Cache Coherence, MANETS etc.*

1. Introduction

Mobile ad hoc networks (MANETS) are self-created and self organized by a collection of mobile nodes, interconnected by multi-hop wireless paths in a strictly peer to peer fashion [1]. Caching is an important part of any on-demand routing protocol for wireless ad hoc networks. In mobile ad hoc network (MANETS) [2],[3],[4] all node cooperate in order to dynamically establish and maintain routing in the network , forwarding packets for each other to allow communication between nodes not directly within wireless transmission range. Rather than using the periodic or background exchange of routing information common in most routing protocols , an on-demand routing protocols is one that searches for the attempts to discover a route to some destination node only when a sending node originates a data packet addressed to the node. In order to avoid the need for such a route discovery to be performed before each data is sent, an on-demand routing protocol must cache routes previously discovered. Such caching then introduces the problem of proper strategies for managing the structure and contents of this cache, as nodes in the network move in and out of wireless transmission range of one another, possibly invalidating some cached routing information.

Several routing protocols for wireless ad hoc networks have used on-demand mechanisms, including temporally-ordered routing algorithm (TORA) [8], Dynamic source Routing protocols (DSR) [5]. Ad hoc on demand distance vector (AODV) [6], Zone routing protocol (ZRP) [7], and Location-Aided Routing (LAR) [9]. For example, in the Dynamic Source Routing protocol [5] in the simplest form, when some node S originates a data packet destined for a node D to which S does not currently know a route, S initiates a new route discovery by beginning a flood a request reaches either D or another node that has a cached route to D, this node then returns to S the route discovered by this request. Performing such a route discovery can be an expensive operation, since it may cause a large number of request packets to be transmitted, and also add latency to the subsequent delivery of data packet that initiated it. But this route discovery may also result in the collection of a large amount of information about the current state of network that may be useful in future routing decision. In particular, S may receive a number of route replies in response to its route discovery flood, each of which returns information about a route to D through a different portion of the network. In high-mobility environment the performance degrades rapidly of this protocol because the

route maintenance mechanism does not locally repair a broken link. Stale cache information could also result in inconsistencies during the route reconstruction phase. In this paper, for efficient searching, we have proposed a generic searching algorithm on associative cache memory organization to faster searching single/multiple paths for destination if exist in intermediate mobile node cache with a complexity $O(n)$ (Where n is number of bits required to represent the searched field).The other major problem of DSR is that the route maintenance mechanism does not locally repair a broken link and Stale cache information could also result in inconsistencies during the route discovery/reconstruction phase. So to deal this, we have proposed an optimized cache coherence handling scheme for on -demand routing protocol (DSR). In this paper, Section 2, we describes the Dynamic Source Routing Protocol (DSR), Section 3, we describe related work, Section 4, we describe associative searching Flowchart, Algorithm and their implementation with example, Section 5, we describe proposed an optimized cache handling scheme and Section 6,we had concluded the paper and future works.

2. Overview of the Dynamic Source Routing Protocols (DSR))

Dynamic source routing protocol (DSR) is an on-demand protocol designed to restrict the bandwidth consumed by control packets in ad hoc wireless networks by eliminating the periodic table-update messages required in the table-driven approach [10]. The major difference between this and other on-demand routing protocols is that it is beacon-less and hence does not require periodic hello packet (beacon) transmission, which are used by a node to inform its neighbors of its presence. The basic approach of this protocol (and all other on-demand routing protocols) during the route construction phase is to establish a route by flooding Route Request packets in the network. The destination node, on receiving a RouteRequest packet, responds by sending a RouteReply packet back to the source, which carries the route traversed by the RouteRequest packet received.

Consider a Source node that does not have a route to the destination. When it has data packets to be sent to that destination, it initiates a RouteRequest packet. This RouteRequest is flooded throughout the network. Each node, upon receiving a RouteRequest packet, rebroadcasts the packet to its neighbors if it has not forwarded already or if the node is not the destination node, provided the packets time to live (TTL) counter has not exceeded. Each RouteRequest carries a sequence number generated by the source node and the path it has traversed. A node, upon receiving a RouteRequest packet, checks the sequence number on the packet before forwarding it. The packet is forwarded only if it is not a duplicate RouteRequest. The Sequence number on the packet is used to prevent loop formations and to avoid multiple transmissions of same RouteRequest by an intermediate node that receives it through multiple paths. Thus, all nodes except the destination forward a RouteRequest packet during the route construction phase.

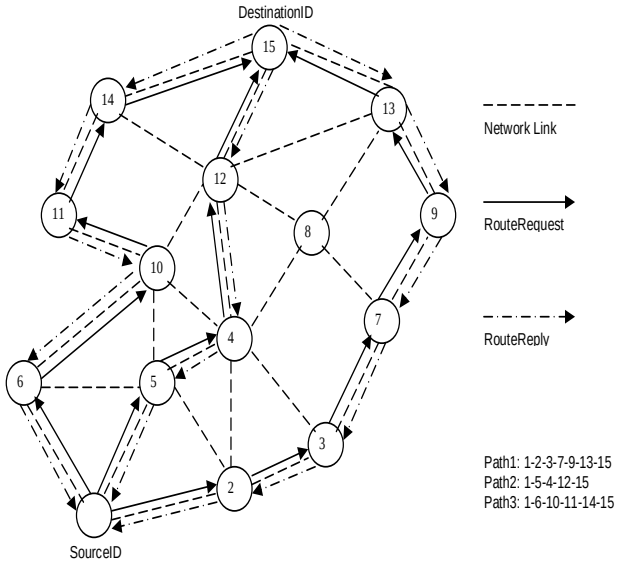

Figure 1. Route establishment in DSR

A destination node, after receiving the first RouteRequest packet, replies to the source node through the reverse path the RouteRequest packet had traversed. In Figure 1, source node 1 initiates a RouteRequest packet to obtain a path for destination node 15. This protocol uses a route cache that stores all possible information extracted from the source route contained in a data packet. Nodes can also learn about the neighboring routes traversed by data packets if operated in the promiscuous mode (the mode of operation in which a node can receive the packets that are neither broadcast nor addressed to itself). This route cache is also used during the route construction phase. If an intermediate node receiving a RouteRequest has a route to the destination node in its route cache, then it replies to the source node by sending a RouteReply with the entire route information from the source node to the destination node.

2.1 Optimizations:

Several optimization techniques have been incorporated into the basic DSR protocol to improve the performance of the protocol. DSR uses the route cache at intermediate nodes. The route cache is populated with routes that can be extracted from the information contained in the data packets that get forwarded. This cache information is used by the intermediate nodes to reply to the source when they receive a RouteRequest packet and if they have a route to the corresponding destination. By operating in the Promiscuous mode, an intermediate node learns about

route breaks. Information thus gained is used to update the route cache so that the active routes maintained in the route cache do not use such broken links. During network partitions, the effected nodes initiate RouteRequest packets. An exponential backoff algorithm is used to avoid frequent RouteRequest flooding in the network when the destination is in another disjoint set. DSR also allows piggy-backing of a data packet on the RouteRequest so that a data packet can be sent along with the RouteRequest.

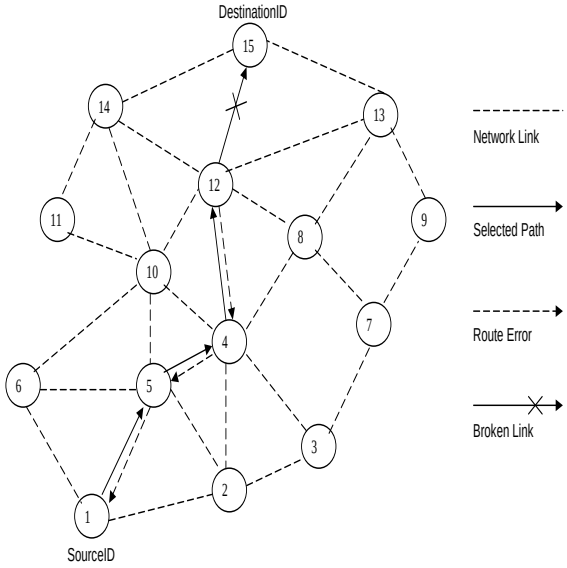

Figure 2. Route maintenance in DSR

If optimization is not allowed in the DSR protocol, the route construction phase is very simple. All the intermediate nodes flood the RouteRequest packet if it is not redundant. For example, after receiving the RouteRequest packet from node 1 from Figure 1,all its neighboring nodes , that is, nodes 2,5, and 6, forward it. Node 4 receives the RouteRequest from both nodes 2 and 5. Node 4 forwards the first RouteRequest it receives from any one of the nodes 2 and 5 and discards the other redundant/duplicate RouteRequest packets. The RouteRequest is propagated till it reaches the destination which initiates the RouteReply. As part of optimizations, if the intermediate nodes are also allowed to originate RouteReply packets, then a source node may receive multiple replies from intermediate nodes. For example, in figure 2, if the intermediate node 10 has a route to the destination via node 14, it also sends the RouteReply to the source node. The Source node selects the latest and the best route, and uses that for sending data packets. Each node packet carries the complete path to its destination. When an intermediate node in the path moves away, causing a wireless link to break, for Example, the link between nodes 12 and 15 in Figure 2, a RouteError message is generated from the node adjacent to the broken link to inform the source node. The source node reinitiates the route establishment procedure. The cached entries at the intermediate nodes and the source node are removed when a RouteError packet is received. If a link breaks due to the movement of the edge nodes (node 1 and node 15), the source node again initiates the route discovery.

3. Related Works.

3.1 Cache data and cache path

The cache data scheme considers the cache placement policy at intermediate nodes in the routing path between the source and destination. The node caches a passing by data item locally when it finds that the data item is popular, i.e., there were many requests for data item, or it has enough free cache space. Since cache data needs extra space to save the data, it should be used prudently. A conservative rule is proposed as follow: A node does not cache the data if all requests for the data are from the same node. However, it uses cooperative caching protocol among mobile node. Each mobile node does not independently perform the caching tasks such as placement and replacement. Cache path is also proposed for redirecting the requests to the cache node. In MANETS, the network the network topology changes fast and thus, the cached path may become invalid due to the movement of mobile nodes [11].

3.2 Neighbor Caching Technique

The concept of neighbor caching (NC) is to utilize the cache space of inactive neighbors for caching tasks. The basic operations of NC are as follow. When a node fetches a data from remote node, it puts the data in its own caching space for reuse. This operation needs to evict the least valuable data from the cache based on a replacement algorithm. With this scheme, the data that is to be evicted is stored in the idle neighbor nodes storage. In the near future if the node needs the data again, it requests the data not from the far remote source node but from the near neighbor that keeps the copy of data. The NC scheme utilizes the available cache space of neighbor to improve the caching performance. However, it lacks the efficiency of the cooperative caching protocol among the mobile nodes [12].

3.3 Node caching schemes

This is a novel approach to constrain route request broadcast based on node caching. The Intuition used is that the nodes involved in recent data packet forwarding have more reliable information about its neighbors and have better locations (e.g., on the intersection of several data routes) than other MANET nodes. The nodes which are recently involved in data packet forwarding are considered as cache nodes, and only they are used to forward route

requests. The modified route request uses a fixed threshold parameter H. The first route request is sent with the small threshold H. When a node N receives the route request, it compares the current time T with the time T(N) when the last data packet through N has been forwarded. If T-H > T(N), then N does not belong to the current cache and , therefore, N will not propagate the route request. Otherwise, if T-H <= T(N), then N is in the node ache and the route request is propagated as usual[13].

3.4 Group Caching

There are some challenges and issue such as mobility of mobile nodes, power consumption in battery , and limited wireless bandwidth when caching techniques are employed in MANETs for data communication .Due to the movement of mobile nodes, MANETs may be partitioned into many independent networks. Hence, the requester cannot retrieve the desired data from the remote server (data source) in another network. The entire data accessibility will be reduced. Also, the caching node may be disconnected from the network for saving power. Thus, the cached data in a mobile node may not be retrieved by other mobile nodes and then usefulness of the cache is reduced. The mobile nodes also decide the caching policy according to the caching status of other mobile nodes. However, the existing cooperative caching in a MANET lack an efficient protocol among the mobile nodes to exchange their localized caching status for caching tasks. In this work a novel cooperative caching scheme called Group caching (GC) which maintains localized caching status of 1 hop neighbors for performing the tasks of data discovery, caching placement, and caching replacement when a data request is received in a mobile node. Each mobile node and its 1 hop neighbors form a group by using the "Hello" message mechanism. In order to utilize the cache space of each mobile node and its 1 hop neighbors form a group by using space of each mobile in a group, the mobile nodes periodically send their caching status in a group. Thus, when caching placement and replacement need to be performed, the mobile node selects the appropriate group member to execute the caching task in the group; this reduces redundancy of cached data objects [14].

Another work is intelligent caching a technique in which, a node not only saves the path discovered during route discovery for itself but also for others who are located close to it. This technique reduces the number of route request packets unnecessarily circulating in the network, when the path it requires is present in its neighborhood.[15]. Authors of [16] in order to share internet contents among mobile users by utilizing low cost wireless connectivity, a content delivery framework with a new content perfecting strategy (AGCS).Another work in which cache management, cooperative caching increase the effective capacity of cooperative caches by minimizing duplications within the cooperation zone and accommodating more data varieties. In this work authors evaluate the performance of the neighbor Group Data caching by using NS2 and compare it with the existing schemes such as Neighbor caching and Zone Cooperative [17].In[18] Authors Proposed epoch numbers, to reduce the problem of cache staleness, by preventing the re-learning of stale knowledge of a link after having earlier heard that the link has broken. In [19] Authors discuss undesirable side effect including cache inefficiencies due to stale paths, and the use of low quality paths even when significantly shorter path become available .

4. An Efficient Associative Search Scheme

Associative memories are mainly used for the faster search and ordered retrieval of large files of records. Many researchers have suggested using associative memories for implementing relational database machines. In this paper, we have proposed a generic searching algorithm on associative cache memory organization to faster searching single/multiple paths for destination if exist in intermediate mobile node cache with a complexity $O(n)$ (Where n is number of bits required to represent the searched field).So for that tabulation of routing records of mobile nodes can be programmed into the cells of an associative memory.

Various Associative Search operations have been classified in to the following categories.

Extreme Search:

Maxima: Find the largest one among a set of records.

Minima: Find the smallest one among a set of records.

Median: Find the median according to a particular ordering.

Equivalence Search:

Equal To: Search is made for a exact match.

Not Equal To: Find all the records which are not equal to given key.

Similar To: Search is made within a masked field.

Proximate To: Find all the records which satisfy a proximate condition.

Threshold Search:

Smaller Than: Find all the records which are strictly smaller than the given key.

Greater Than: Find all the records which are strictly greater than the given key.

Not Smaller Than: Find all the records which are equal to or greater than the given key.

Not Greater Than: Find all the records which are equal to or less than the given key.

Adjacency Search:

Nearest Below: Find nearest record in which key is smaller than the given key.

Between limit Search

$[x, y]$: find all records within the closed range.

$\{Z \mid X \le Z \le Y\}$

(X,Y) : Find all records in the open range.

$\{Z \mid X < Z < Y\}$

$[X,Y)$: Find all records within in the range.

$\{Z \mid X \le Z < Y\}$

$(X,Y]$: Find all records within in the range

$\{Z \mid X < Z \le Y\}$

Ordered Search:

Ascending Sort: List all records in the ascending order.

Descending Sort: List all records in the descending order.

Table 1.List of Abbreviations:

C	n bit comparative register
M	n bit masking register
$I(0)$ and $T(0)$	$I(0)$ and $T(0)$ are Index and Temporary $N*1$ bit registers initially set.
N	Number to be searched
K	The Index within the field, where $1 \le k \le f$
J	The index for successive bit slices, where $1 \le j \le m$
R_i and S_i	The reset and set signals of flip flop I_i are denoted as R_i and S_i
S	The starting bit address of a field, where $1 \le s \le n$
f	The field length in bits.
i	The index for different bit slices, where $1 \le i \le n$
B_{ij}	Represent jth bit position of ith memory word

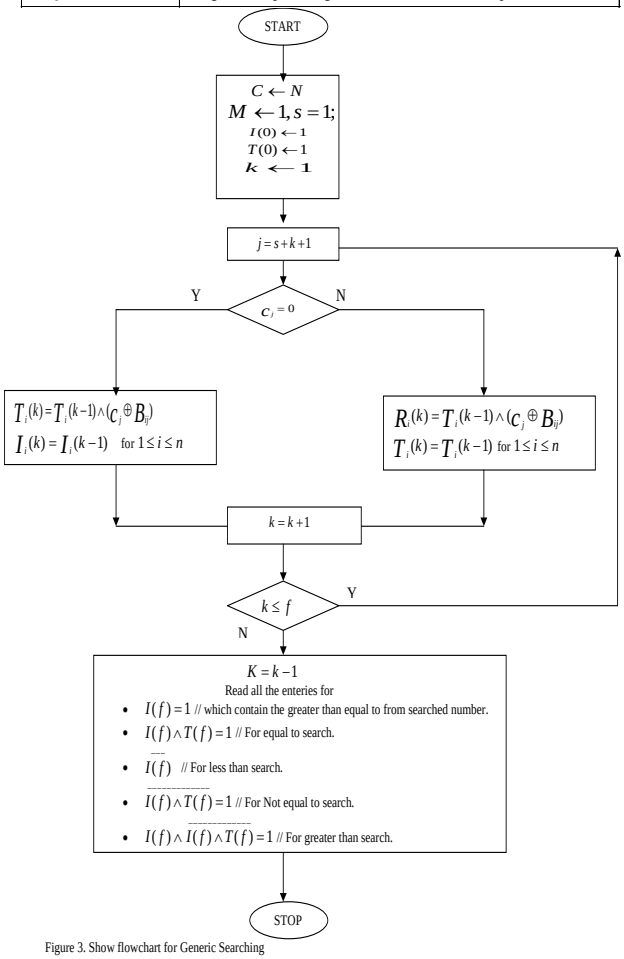

Figure 3. Show flowchart for Generic Searching

4.1 Generic search algorithm for search number from Associative Cache memory Organization

Generic-search (C,N,n,f,s,Si,Ri,m,k,I(0),T(0))

$Step1: C \leftarrow N$

$Step2: M \leftarrow 1, s = 1;$ //M is n bit masking register & all its bits are 1 (to search all the bits of register C)

$Step3: I(0) \leftarrow 1$ // I (0) is an N*1 bit Index Register (Initially set)

$Step4: T(0) \leftarrow 1$ // T (0) is an N*1 bit Temporary Register (initially set)

$Step5: k \leftarrow 1$ // k is used for two purpose
I) To point the next bit position in the field.
II) To represent stage number.

$Step6: j = s + k + 1$

$if\ (C_j = 0)$

$T_i(k) = T_i(k-1) \wedge (C_j \oplus B_{ij})$

$I_i(k) = I_i(k-1)$ for $1 \le i \le n$

else

$R_i(k) = T_i(k-1) \wedge (C_j \oplus B_{ij})$

$T_i(k) = T_i(k-1)$ for $1 \le i \le n$

$Step7: k = k + 1$

$if\ (k \le f)\ then\ go\ to\ Step6$

else

$k = k - 1$

Read the Index register $I(f) = 1$ their set bits positions contain the number greater than equal to from searched number.

Read all the enteries of $I(f) \wedge T(f) = 1$ its set bits position for equal to search.

Read all the enteries of $\bar{I}(f) = 1$ its set bits position for less than search.

Read all the enteries of $\overline{I(f) \wedge T(f)} = 1$ its set bits position for Not equal to search.

Read all the enteries of $I(f) \wedge \overline{I(f) \wedge T(f)} = 1$ its set bits position for greater than search.

4.2. Implementation of Generic Search Algorithm with example.

In associative memory time required to find an item stored in memory can be reduced considerably because stored data can be identified by the content of the data

itself rather than by the address. An associative memory is also known as content addressable memory (CAM).The block diagram of an associative memory is shown in figure. It consists of a memory array for 4 words with 4 bits per word. The comparative register C hold the item that you want to search and masking register M are also 4bits. The masking register provides a mask for choosing a particular field. The entire bits of register C are compared with each memory word if the masking register contains all 1's.There are also two 4*1 bit size index (I) and temporary (T) which are initially set. In this section, Implementation of Generic search Algorithm on example is shown below.

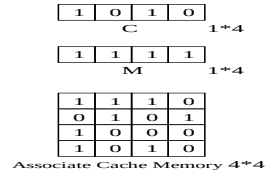

Figure 4. Shows Associate Memory

I (0)	T (0)
1	1
1	1
1	1
1	1

$step1: k = 1, \quad j = s + k - 1 = 1$ and $f = 4;$

$step2: C_1 = 1$ \\There fore the Index Register effected and no change in Temporary Register.

$R_1(1) = (1) \wedge (1 \oplus 1) = 0$ i.e. No Change
$R_2(1) = (1) \wedge (1 \oplus 0) = 1$ Reset
$R_3(1) = (1) \wedge (1 \oplus 1) = 0$ i.e. No Change
$R_4(1) = (1) \wedge (1 \oplus 1) = 0$ i.e. No Change

So the value of I (1) & T (1)

I (1)	T (1)
1	1
0	1
1	1
1	1

$step1: k = 2, \; j = s + k - 1 = 2$

$step2: C_2 = 0$ \\There fore the Temporary Register effected and no change in Index Register

$T_1(2) = (1) \wedge \overline{0 \oplus 1} = 0$
$T_2(2) = (1) \wedge \overline{0 \oplus 1} = 0$
$T_3(2) = (1) \wedge \overline{0 \oplus 0} = 1$
$T_4(2) = (1) \wedge \overline{0 \oplus 0} = 1$

So the value of I (2) & T (2)

I (2)	T (2)
1	0
0	0
1	1
1	1

$step1: k = 3, \; j = s + k - 1 = 3$

$step2: C_3 = 1$ \\There fore the Index Register effected and no change in Temporary Register.

$R_1(3) = (0) \wedge (1 \oplus 1) = 0$ i.e. No Change
$R_2(3) = (0) \wedge (1 \oplus 0) = 0$ i.e. No Change
$R_3(3) = (1) \wedge (1 \oplus 0) = 1$ Reset
$R_4(3) = (1) \wedge (1 \oplus 1) = 0$ i.e. No Change.

So the value of I (3) & T (3)

I (3)	T (3)
1	0
0	0
0	1
1	1

$step1: k = 4, \; j = s + k - 1 = 4$

$step2: C_4 = 0$ \\There fore the Temporary Register effected and no change in Index Register.

$T_1(4) = (0) \wedge \overline{0 \oplus 0} = 0$
$T_2(4) = (0) \wedge \overline{0 \oplus 1} = 0$
$T_3(4) = (1) \wedge \overline{0 \oplus 0} = 1$
$T_4(4) = (1) \wedge \overline{0 \oplus 0} = 1$

So the value of I (4) & T (4)

I (4)	T (4)
1	0
0	0
0	1
1	1

Read the Index register $I(4) = 1$ its set bits position contain the number greater than equal to from searched number

Read all the enteries of $I(4) \wedge T(4) = 1$ its set bits position for equal to search.

Read all the enteries of $\overline{I(4)} = 1$ its set bits position for less than search.

Read all the enteries of $\overline{I(4) \wedge T(4)} = 1$ its set bits position for Not equal to search.

Read all the enteries of $I(4) \wedge \overline{I(4) \wedge T(4)} = 1$ its set bits position for greater than searched number.

5. Proposed an Optimized Cache Coherence Handling Scheme

Ad hoc networks (MANETS) are self-created and self organized by a collection of mobile nodes, interconnected by multi-hop wireless paths in a strictly peer to peer fashion. Caching is an important part of any on-demand routing protocol for wireless ad hoc networks. In mobile ad hoc network (MANETS) all nodes cooperate in order to dynamically establish and maintain routing in the network, forwarding packets for each other to allow communication between nodes not directly within wireless transmission range. For that Several optimization techniques have been incorporated into the basic DSR protocol to improve the performance of the protocol. DSR uses the route cache at intermediate nodes. The route cache is populated with routes that can be extracted from the information contained in the data packets that get forwarded. This cache information is used by the intermediate nodes to reply to the source when they receive a RouteRequest packet during Route Discovery Phase.

Due to presence of private cache for each mobile node in an ad hoc network necessarily introduces problems of cache coherence, which may result in data inconsistency. Clearly, the cache coherence problem cannot be solved by a memory Write-through policy. If a Write-through policy is used, the main memory location is updated, but the possible copies of the routing information in other caches are not automatically updated by the write-through mechanism. So "Write-through: is neither necessary nor sufficient for cache coherence. For that in this paper, we have proposed a dynamic coherence check scheme for

cache coherence in routing table of mobile nodes for MANETs.

In this existing scheme, called dynamic coherence check, multiple copies are allowed. However, whenever a mobile node moves and modifies routing information in its local cache, it must check the other caches to invalidate possible copies. This operation is referred to as a *cross-interrogate* (XI). In other words, when a mobile node writes into a shared block X in its cache, the node sends a signal to all the remote caches to indicate that the "data at memory address X has been modified." At the same time, it writes through memory. Note that, to ensure correctness of execution, a mobile node which requests an XI must wait for an acknowledge signal from all other mobile nodes before it can complete the write operation. The XI invalidates the remote cache location corresponding to *X* if it exists in that cache. When the other mobile node references this invalid cache location, it results in a cache miss, which is serviced to retrieve the block containing the updated information. In this approach, for each write operation, $(n - 1)$ XIs result, where n is the number of mobile nodes. Note that the two sources of inefficiency for this technique are the necessity of a write-through policy, which increases the network traffic, and the redundant cache XIs which are performed. In the latter case, a cache is purged blindly whether or not it contains the data item *X*.

In our proposed scheme, our objective is to optimize cache coherence handling scheme .In this scheme, we focus on a more refined technique filters the *cross-interrogate* (XI) requests before they are initiated on reactive routing protocol DSR for mobile ad hoc network. For that, we have ad hoc network in which every mobile node having own local cache and there is one mobile node having the centralized shared main memory. This main memory contains the memory control element (MSC) maintains a central copy of the directories of all the caches. We will elaborate on a similar scheme called the presence flag technique**,** which assumes a write-back main memory update policy. There are two central tables associated with the blocks of main memory (MM) as shown in Figure5. The first table is a two-dimensional table called the Present table. In this table, each entry $P[i,c]=1$, contains a *present* flag for the *i*th block in MM and the cth cache. If $P[i,c]=1$, then the cth cache has a copy of the *i*th block of MM, otherwise it is zero. The second table is the *Modified* table and is one-dimensional. In this table, each entry M[i] contains a *modified* flag for the *i*th block of MM. If $M[i]=1$, it means that there exists a cache with a copy of the *i*th block more recent than the corresponding copy in MM. The Present and Modified tables can be implemented in a fast random-access memory. The philosophy behind the cache coherence check is that an arbitrary number of caches can have a copy of a block, provided that all the copies are identical. They are identical if the Mobile node associated with each of the caches has not attempted to modify its copy since the copy was loaded in its cache. We refer to such a copy as *read only* (RO) copy. In order to modify a block copy in its cache, a mobile node must own the block copy with *exclusive read only* (EX) or *exclusive read-write* (RW) access rights. A copy is held EX in a cache if the cache is the only one with the block copy and the copy has not been modified. Similarly, a copy is held RW in a cache if the cache is the only one with the block copy and the copy has been modified. Therefore, for consistency, only one mobile node can at any time own an EX or RO copy of a block.

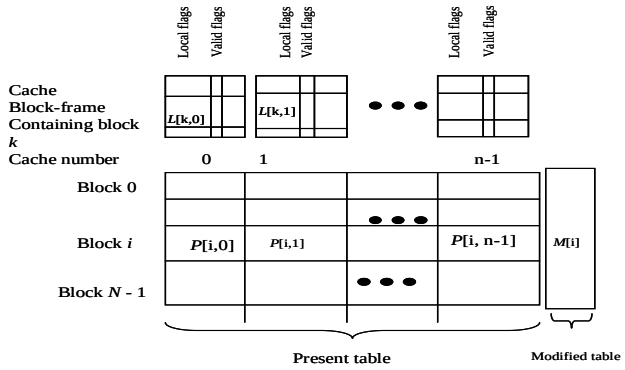

Figure 5. Organization of flags for dynamic solution to cache coherence

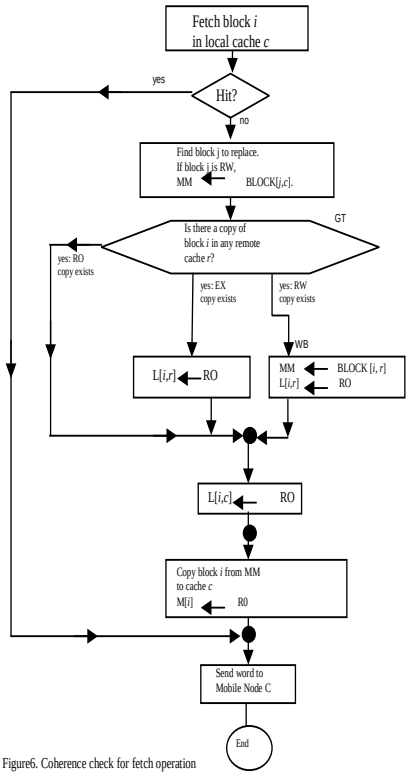

Figure6. Coherence check for fetch operation

GT: possible access to global table
WB: possible MM update
MM: main memory
M[*i*]: modified bit of block *i* in GT
MM :BLOCK [*i,r*]: update MM with modified copy of *i* in cache *r*
L[*i,r*]: state flag of block *i* in cache *r*

To enforce the cache consistency rule, local flags are provided within each cache in addition to the global tables. A local flag $L[k,c]$ is provided for each block *k* in cache *c*. This flag indicates the state of each block in the cache. A block in a cache can be in one of three states: RO, EX, or RW. Figure 6 shows the flowchart for the HIT or MISS when mobile node c fetches the block ith from their local cache.

As long as the copy of block *i* remains present in the cache, mobile node *c* can fetch it without any consistency check. If mobile node *c* attempts to store into its copy of block *i*, it must ensure that all other copies (if any) of block *i* are invalidated. To do this, the global table is consulted. It should indicate the mobile node caches that own a copy of block *i*. The modified bit for block *i* is updated in the global table to record the fact that mobile node *c* owns block *i* with RW access rights. Finally, the local $L[k,c]$ flag is set to RW to indicate that the block is modified.

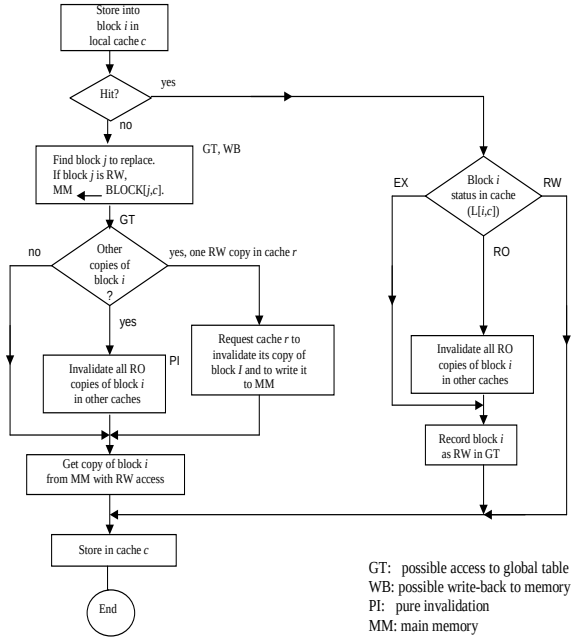

Figure 7. Coherence check for store operation

The flowchart for a store is shown in Figure 7.In this implementation, a block copy in a cache is invalidated whenever the cache receives a signal from some other mobile node attempting to store into it. Moreover, a cache which owns an RW copy may receive a signal from a remote cache requesting to own an RO copy. In this case, the RW copy's state is changed to RO.

6. Conclusions and Future work

In this paper, we have proposed a generic searching algorithm on associative cache memory organization to enhance the searching of single/multiple path for destination, if exist, in intermediate mobile node cache with a complexity $O(n)$ (Where n is number of bits required to represent the searched field).The proposed algorithm reduces the route discovery over head. We have proposed an optimized dynamic coherence check scheme to reduce the number of *cross-interrogate* (XI) signal sends to different mobile node caches resulting in the reduction of routing over head on DSR protocols. In future work, the focus will be on the effect of different page replacement polices [FIFO, LRU etc.] of cache and to reduce the number of conflicts that occurs when concurrent access to the global table occurs.

References

[1]Jeremy Pitt, Pallapa Venkatarram, and Abe Mamdani, "QoS Management in MANETS Using Norm-Governed AgentSocieties" ESAW 2005, LNAI 3963, Page(s): 221- 240, 2006.

[2]Internet Engineering Task Force MANET Working Group. Mobile Ad hoc networks (Manet) Charter Available at http:// www.ietf.org/html.charters/manet-charter.html.

[3]Asis Nasipuri, Mobile Adhoc networks,Department of Electrical and Computer Engineering, The university of North Carolina at charlotte.Charlotte,NC 28233-0001.

[4] C.E. Perkins, Ad hoc networking, Addison-Wesley, 2001.

[5] D.Johnson, Rice University; Y. Hu,UIUC and D.Maltz, Microsoft Research The Dynamic Source Routing Protocol (DSR) for mobile ad hoc networks for IPV4, February 2007 http://www.ietf.org/rfc/rfc4728.txt.

[6] C. Perkins and S. Das, Ad hoc On Demand Distance Vector (AODV) Routing IETF, Internet Draft, draft-ietf-manet-aodv-13,RFc 3561, 17,February -17- 2003.

[7] Z. Haas, M. Pearlman nad P. Smar, Zone routing protocol (ZRP), Internet Draft, Internet Engineering Task Force , Jan 2001,http://www.ietf.org/internet-drafts-ietf-manet-zoneierp-00.txt.

[8] S. Bradner , temporally-ordered routing algorithm (TORA) Routing IETF, Internet draft-ietf-manet-tora-spec-04.txt,RFC 2026,July 2001.

[9] Y. Kuo, and N.H. Vaidya, " Location –Aided Routing (LAR) Mobile Ad Hoc Networks," In proceedings of the International Conference on Mobile Computing and Networking (MobiCom'98),Oct.1998.

[10] David .B Johnson, David. A. Maltz, and Josh Broch, " Dynamic Source Routing protocol for Multihop Wireless Ad Hoc Networks," In Ad Hoc Networking, edited by Charles E. Perkins, chapter 5, pages 139-172. Addison-Wesley, 2001.

[11] LiangZhong Yin and Guohong Cao, " Supporting Cooperative caching in ad hoc networks ," IEEE Transactions on Mobile computing, Vol. 5, Issue 1,pages 77-89,jan. 2006.

[12] Joonho Cho, Seungtaek Oh, Jaemyoung Kim, Hyeong Ho Lee, and Joonwon Lee, " Neighbor caching in multi-hop

wireless ad hoc networks," IEEE Communications Letters, Vol 7,issue Nov. ,pages 525-527,2003.
[13] Sunlook Jung,Nisar Hundewale, and Alex Zelikovsky, " Node Caching Enhancement of Reactive Ad hoc routing Protcols," IEEE Wireless Communications and Networking Conference, 2005.
[14] Yi-Wei Ting and Yeim-Kuan Chang, " A novel Cooperative Caching Scheme for Wireless Ad hoc Networks; GroupCaching," International Conference on Networking Architecture and storage ,NAS 2007.
[15] Shobha.K.R., and K. Rajanikanth "Intelligent caching in On-Demand Routing Protocol for Mobile Adhoc Networks" World Academy of Science Engineering and Technology 56 pages 413-420,2009.
[16] Yaozhou Ma, M. Rubaiyat Kibria, and Abbas Jamalipour "Cache-based Content Delivery in Opportunistic Mobile Ad hoc Networks" IEEE "GLOBECOM", 2008.
[17] Mrs. K. Shanmugavadivu and Dr. M. Madheswaran "CachingTechnique for Improving Data Retrieval Performance in Mobile Ad Hoc Networks" In International Journal of Computer Science and Information Technologies (IJCSIT),Vol. 1(4),pages 249-255,2010.
[18]Yih-Chun Hu and David B.Johnson " Ensuring Cache Freshness in On-Demand Ad hoc Network Routing Protocols" In POMC'02, October 30-31, Toulouse, France ,2002.
[19] Nikhil I. Panchal and Nael B. Abu-Ghazaleh "Active Route Cache Optimization for Ad hoc Networks" In Infocom 2002.

First Author Rajneesh Kumar Gujral is working as Assoc. Professor Department of Computer Engineering, M.M Engineering College, M.M.University Mullana, Ambala. He obtained his BE (Computers) in 1999 unit SLIET Longowal from Punjab Technical University(PTU), Jalandhar. He also obtained his MTECH (IT) in 2007 from University School of Information Technology, GGSIP University Delhi. He has about 10 publications in journals and International Conferences to his credit. His current research interest includes Wireless communications which include mobile, Adhoc and sensor based networks, Network Security and computer communication networks etc.

Second Author Dr. Anil Kumar Kapil is working as Professor. & Principal M. M. Institute of Computer Technology and Business Management, M. M. University Mullana, Ambala, India. He obtained his Ph.D. (Computer Science & Engineering) in 2007. He has about 25 publications in journals and International Conferences to his credit. His current research interest includes Wireless communications which include mobile, Adhoc and sensor based networks, computer communication networks, Distributed networks and Concurrency control etc.