American Journal of Applied Sciences 9 (8): 1307-1315, 2012
ISSN 1546-9239

Cluster Based Data Consistency for Cooperative Caching over Partitionable Mobile Adhoc Network

[1]Kuppusamy, P. and [2]B. Kalaavathi
[1]Department of Computer Science and Engineering,
Vivekanandha College of Engineering for Women, Namakkal, India
[2]Department of Computer Science and Engineering,
K.S.R Institute for Engineering and Technology, Namakkal, India

Abstract: Problem statement: Data availability and consistency are foremost issues in Mobile Ad hoc Networks due to the absence of permanent infrastructure. Cooperative caching addresses the data availability issue through coordinating the mobile nodes and sharing the cache copies among them. In the meantime, the mobile node must ensure the staleness of cache copies. The consistency maintenance resolves the staleness of the data among the source and caching node. Due to addition of more mobile nodes the network size is increased and it leads to increase the caching nodes. The mobility and disconnections causes additional overhead, latency and reduces the data delivery success ratio while updating the cache copy from the source. **Approach:** This study proposed Adaptive Push and Pull Algorithm for Clusters (APPC) and Cluster Based Data Consistency (CBDC) approach to address the consistency requirements and maintenance in mobile ad hoc network. **Results:** The CBDC satisfy the consistency requirements in partitioned clusters. The source node transmits the updated data through the cluster heads to the caching nodes. The APPC ensures the validity of cache copy by threshold Time-to-Live (TTL). Thus it provides efficient valid data accessibility in mobile ad hoc network. **Conclusion:** The simulation results shown that this proposed approach increases packet success ratio and reduces the delay and overhead when compared with the existing approaches Flexible Cache Consistency (FCPP) maintenance and Cluster Based Cooperative Caching Technique (CBCCT) for increasing number of nodes and speed respectively.

Key words: Time-To-Live (TTL), invalidation, update delay, cooperative caching, cache copy, Cluster Based Data Consistency (CBDC), Adaptive push and pull for clusters, MANET

INTRODUCTION

Mobile Ad Hoc Networks (MANETs) consist self governing Mobile Nodes (MNs) with dynamic infrastructure and multi-hop wireless links. The previous researches have primarily focused on routing and MAC protocols in MANET. Although routing and MAC protocols having important issue such as efficient data access in MANET. Moreover, the MANET contains some limitations like battery energy constraint, limited bandwidth, unpredictable signal propagation, mobility and unreliable wireless links. This causes frequent disconnection in the network that makes issues in data availability and accessibility. Cooperative caching is an efficient way to tackle these issues and improve the system performance in terms of energy, query latency, data delivery and overhead. MNs are cooperating with each other to share the data that reduces remote server's workload and communication channel bandwidth. Due to rapid progress in wireless network, MANETs are not only used in military operations and also used in commercial and industrial applications like news, traffic information, cricket score updates and stock market. In cooperative caching, the accessing shared data is widely cached in the caching nodes. The shared cache copy is not a static, it is modified and updated in the source during its lifetime. The modified and/or updated data in the source must be replicated to cache copy in the caching node. Since data have cached in many caching nodes, it requires consistency approaches to ensure that all cache copies are consistent with source. Thus maintaining cache consistency is a challenging issue in the mobile environment. The novel consistency approach is predominantly proposed to handle the consistency among the cache copy in caching node and source.

Corresponding Author: Kuppusamy, P. Department of Computer Science and Engineering,
Vivekanandha College of Engineering for Women, Namakkal, India

Motivation: In fast development of the mobile communication, the MN retrieves the required data from the remote located source node. The MNs frequently change its location during its data transmission due to network dynamism and mobility. MNs cannot retrieve the required data from the remote source at all time in the huge network. Hence, MNs caches the accessed data from remote source to share with its neighbors. This cache copy improves data availability in the network. But, the query latency and overhead have decreased drastically in the huge network due to numerous caching nodes and also invalidation of data takes long time to update the cache copies in caching nodes from the source. The source also must ensure the consistency of cache copies in the caching nodes. It motivates the researcher to make exploration on maintain the consistency among source and caching node over huge MANET.

Problem identification and proposed solution: The most of the previous research works on cache consistency did not mention any specific approach to handle the consistency among data source and caching nodes with mobility and disconnection in huge MANET. Hence there is a necessity to design an effective consistency approach to maintain the consistency to handle disconnections in huge network.

In this study, we propose to develop an Adaptive Push and Pull Algorithm for Clusters (APPC) and Cluster Based Data Consistency (CBDC) approach to address the consistency maintenance in partitionable MANET. The proposed algorithms can alleviate the consistency issues with mobility and disconnection in huge MANET.

Literature review: There are many researches on the caching and consistency maintenance algorithms for distributed environments such as Web, P2P systems, database and mobile wireless network. However, these approaches cannot be directly applied in MANET due to dynamic topology, limited bandwidth, mobility, energy constraint (Cao *et al*., 2005). Traditional consistency control approaches are push and pull schemes. The Push-based schemes are suitable for stable network which guarantees for nodes which are online and reachable from the source at all time. However, these schemes have low query latency and cannot solve the disconnection problem. The caching nodes cannot receive the invalidation messages due to disconnections that results sharing of the stale data upon reconnection. Pull-based schemes are more suitable for dynamic networks which cause high communication overhead in message flooding and caching nodes consume much battery energy. The conventional cache status maintenance approaches are Stateful (SF) and Stateless (SL). In SF (Cao, 2002), data source avoids redundant broadcast flooding in the network. The data source aware about all cache copies in the caching nodes. Hence, it requires a large and complicated database. In SL (Imielinski and Barbara, 1994), data source not aware of cache copies status and simple to handle and implement. However it causes more overhead due to floods more redundant messages.

The most important consistency level approaches are Strong Consistency (SC) and Weak Consistency (WC). In SC, the cache copies in the caching nodes are up-to-date at all time. In WC, consistency of cache copies is maintained among source and caching node, but not provide assurance on the deviation between them. The Delta Consistency (DC) satisfies the maximum acceptable deviation between the source and the cache copy. CBDC proposed to provide the consistency requirements between SC and WC. Each cache copy associates with Time-to-Live (TTL) value. It provides acceptable deviation among the source and cache copies through cluster head. Many consistency algorithms have been proposed for consistency maintenance.

Cao *et al*. (2004) have presented simple weak consistency model in which cache copy associate with Time-To-Refresh value. In this model, the request forwards to the data source if TTR is expired in caching node. It causes long query delay. Duvvuri *et al*. (2003) presents a new lease approach to provide SC in that the source data is not modified without prior notification as long as the lease is valid. Huang *et al*. (2010a) have proposed predictive consistency control initiation scheme to provide WC in that source node proactively propagates updates to the caching nodes. But the source node does not know whether the caching nodes require data updates and it also induces round trip cost. Cao *et al*. (2007) have proposed relay peer-based cache consistency to provide DC based on TTR value. The relay peers selected by data source from stable, high energy nodes and push updates to these relay peers periodically. Other caching nodes receive the updated data from relay peers in a pull scheme. This scheme increases the load on the relay peer and pull request broadcasting consumes bandwidth and energy. Feeney (2001) has presented energy consumption of MANET routing protocols. Xie *et al*. (2007) have proposed dynamic tree based consistency in which data updated through a binary tree. However, this model makes additional overhead during updating the tree due to node's mobility. Jing *et al*. (1997) have presented a Bit-Sequence (BS) approach that uses a hierarchical structure of binary bit sequences to represent invalidations for long disconnections. But, data update rate is not high. Li *et al*. (2007) have presented cache

invalidation strategies which reduce latency, but bandwidth cost is high. Huang *et al*. (2006) have proposed Predictive Caching Consistency algorithm based on the online predictions of data updates and queries. But, these schemes can offer only SC or WC. Li *et al*. (2009) have offered probabilistic cache consistency model to ensure the validity of cache copy with neighbors cache copy instead of data source forever. But it makes unnecessary invalidation when neighbors copy is stale. Kuppusamy *et al*. (2012) have proposed Cluster Based Cooperative Caching Technique (CBCCT) based on mobility and connectivity to improve the data availability over MANET. The cluster member caches data in local cache and updates with its corresponding CH's global cache to share with neighbor clusters. But, the consistency maintained by updating source data periodically to caching nodes. It leads to additional overhead. Artail *et al*. (2008) have proposed cooperation-based database caching system in which MNs cache the submitted queries as indexes. It provides better hit ratios and smaller delays but at the cost of a bandwidth consumption is slightly higher. Huang *et al*. (2010b) have proposed flexible cache consistency algorithm to minimize consistency cost and ensure the consistency based on probability of data validity. It provides SC, WC and DC. However, this scheme can not satisfy the consistency requirements when source data widely cached in unstable network connection.

However none of previous research algorithms provides to handle the consistency issues with mobility and disconnections in huge MANET. This study aims to provide an algorithm to partition the huge network into clusters, share the data among neighbor clusters through Cluster Head (CH) and maintain the consistency between the data source and caching nodes.

Cluster based data consistency: In this model, the MNs cache the data while accessing the data from the source. This caching MN can directly serve the data to its neighbors queries by Cluster based Data Consistency. Whenever the cache copy is accessed by its neighbor MNs within the cluster or neighbor clusters cache copy access rate is computed. When cache copy access rate is greater than the access rate threshold it considered as frequently required data.

When data is updated in source, it sends the invalidation report to the caching nodes CH. The CH replies acknowledgement to the data source immediately and also intimate to its caching nodes about data being updated. Thus the SC is maintained and requesting neighbor need not be waited for long time. The CH maintains the caching nodes information with cache copy's TTL. When TTL of the cache copy is decreasing lower than threshold, the CH initiating the RENEW message before TTL expires. Thus the WC consistency is maintained by pull approach. Assume that the CH, caching nodes and the data source have synchronized clocks.

Let V^t represent the version number of data D_i at the source node and C^t_k represent the cache copy at node k at time t. The initial version number V^t is zero at the data creation time. It is incremented for each consecutive update in the source. This updated version of D_i, send to all caching nodes through their CH. Hence, CH aware of its member's cache copy information with TTL and it informs to their neighbor CHs. When cache copy D_i in the caching node is not stale more than acceptable deviation (δ) time with source, it ensures DC. Maximum acceptable stale time is:

$$C^t_k = V^{t-\tau}{}_k\,; \quad \forall t, \forall k, \exists \tau, 0 \le \tau \le \delta$$

The difference in the values among source and cache copy versions is bound by acceptable deviation:

$$\delta \forall t, \left| C^t_k - V^{t-\tau}_k \right| < \delta$$

Overview of cluster based caching:

Clustering pattern: The cluster is configured based on the spatio-temporal stability of MNs (Kuppusamy *et al*., 2012). The MNs send Hello messages to the neighbors with ID, energy level. The MN with greater energy selected as a CH by neighbors which assigned as a cluster members. Then data are stored in the source MN. MNs send the beacon massage to its CH. The beacon message consists Cluster ID (CID), Data (D) id with TTL, Received Signal Strength (RSS). Then CH replicates its content to other CHs and also collects other CHs contents. If the CH energy level is decreased than cluster member energy, then cluster member is assigned as CH and give up all information to the new CH.

Mobility and disconnections: The movement sequences of MNs can observe from the RSS at regular intervals. These observations estimate the mobility locations of MNs and strength of the connectivity among connections. Thus mobility and connectivity is computed through RSS. The MN may disconnect from the network due to energy. The CH does not receive reply from the MN during that period. Hence, CH receives the updates from the source and keeps in its cache for a short time. The MN sends the cache check invalidation request to its CH upon reconnection. Then CH transmits the updated data. When MN does not reconnect for long time, CH deletes the data. Thus CH resolves the disconnection issue in clusters.

Data query processing: Whenever MN requires the data D, it checks in Local Cache Table (LCT).

Otherwise, it sends request to its CH_2 as in Fig. 1. CH_2 checks in its LCT for the required data. If data is in LCT, it sends reply to its home cluster caching node. If data is not in LCT, CH_2 checks in the GCT (Global Cache Table) about neighbor clusters. If data is there, it sends the query to corresponding cluster CH_1's caching node. Then neighbor CH_1 sends the reply to the MN through its CH_2. MN caches the data in its LCT, after received from neighbor cluster.

Energy model: The energy of MNs have calculated after establishment of the communication between the nodes in the network. According to Kuppusamy *et al.* (2012) energy consumption of MNs for transmitting, receiving k-bits of data at distance d is given as in Eq. 1:

$$E_c = E_{ele}. k + \varepsilon_{amp}. k.d^2 \quad (1)$$

δt is duration of time while node acts as CH. The total energy consumption (E_t) of CH computed in Eq. 2 as:

$$E_t = E_c.\delta t \quad (2)$$

The residual energy E_R of CH is computed in Eq. 3 from initial energy (E_i) and total energy consumption (E_t) as:

$$E_R = E_i - E_t \quad (3)$$

The MN with maximum energy is nominated as a CH. If the E_R CH is decreased than any of its cluster member, then cluster member is nominated as new CH.

Cache consistency requirements: The CBDC approach must satisfy the consistency level, consistency control and data update delay.

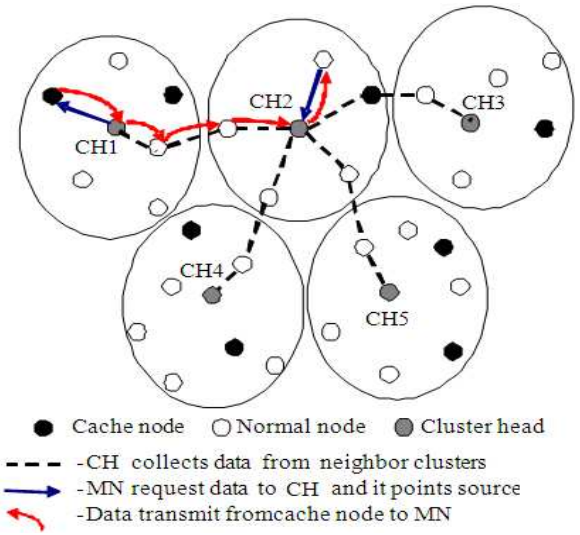

Fig. 1: Data query and retrieval in cluster

Consistency level: The cache must provide all kind of consistency levels SC, WC to the clients with minimum cost. The consistency maintenance cost is based on the time. This issue addresses by CBDC model and satisfies all consistency levels.

Consistency control: The most widely used consistency control approach is the cache copy associate with TTL value. The TTL values of cache copies in the nodes to be renewed from the source node while the TTL values have expired. Once the source node updates the data, it needs to update its cache copies in the caching nodes. Hence it initiates push invalidation report to the caching nodes. The caching nodes must reply the acknowledgement to source node due to network dynamism. Both schemes have mutually used in APPC.

Update delay: The previous schemes mostly focus on how cache consistency should be maintained after the source node has updated the source data. In such schemes, the source can directly update the data without considering consistency maintenance with caching nodes. However, in many cases, the source node can wait for certain time before updating the source data, as in adaptive Lease protocol. The update delay can be utilized to further decrease the consistency maintenance cost. The source node waits for some predefined delay to update the source data in small ad hoc network with stable connection as in flexible cache maintenance. However, it does not provide on how long the source node needs to wait before updating the source data in large size dynamic network. It does not also provide the update delay when MNs in mobility in large network. The APPC provides minimum update delay using by CH.

Adaptive Push and Pull algorithm for Clusters (APPC):

Overview: The proposed APPC satisfies the consistency maintenance cost in cluster based MANET. Each cache copy is associated with TTL which is computed based on acceptable time deviation δ among the source and caching nodes. The caching nodes satisfy their neighbor queries when TTL is not expired. For data update, the source sends an Invalidation Report (INV_REP) to the CHs in the network. The CHs reply Invalidation Report Acknowledgement (INVREP_ACK) to the source. In mean time, CHs intimate about INV_REP to their caching nodes in the cluster. Hence, caching node is not serve the data to neighbors request during that period. The source updates the data after received INVREP_ACK from all CHs. Otherwise, the source waits up to tolerable minimum update delay.

Algorithm.1 APPC on a Source:

```
while (source ready to update data)
{
 send an INV_REP to all CHs with time T
 CHs send INVREP_ACK upon receiving INV_REP.
if (T > 0) && (Data source node receives INVREP_ACK from all CH) then
 update the source data
 push to caching nodes through CH
else if (TTL < t_m) then
 wait for minimum update delay t_m=0
 update the source data
 push to caching nodes through CH
end if
}
end while
while ( TTL < TTL_th)
{
 source receives RENEW request from CH
if (source data updated multiple times among past and current renewal requests) then
        TTL = TTL x m
 send updated data to caching node with new TTL
else if (no data update) then
        TTL = TTL x (1 +f)
 end if
}
end while
```

For renew the data, CH maintains cache copy information associated with their TTL, query access rate. When TTL decreased out of threshold TTL (TTL_{th}), the CH renews the TTL of corresponding cache copy in advance based on access rate. CH knows about the access rate of renewal data based on the neighbors' request. Thus caching nodes need not to wait up to TTL is expired to renew and also can serve the data to neighbors continuously. Hence, the delay reduced at TTL renewal. If the cache copy is not accessed by neighbors for long time then its TTL is not renewed by CH. Thus the overhead is reduced by avoiding unnecessary TTL renewal.

Algorithm.2 APPC on Cluster Head:

```
while (CH receiving INV_REP from source node with T)
{
 if (T>0)
   send INVREP_ACK to the data source node
   send about INV_REP to the corresponding cache node
   ready to receive updated the data
 if (data received with new TTL) then
 send the updated data with new TTL to caching node
 end if
 end if
}
end while
//Upon TTL of cache copy
if (TTL < TTL_th) then
  send the RENEW message to source node for proactive renewal
if (cache copy not updated) then
 update the new TTL for cache copy
 send new TTL to caching node
else if ( cache copy has updated) then
 update cache copy with new TTL
 serve the cache query with out delay
end if
end if
```

APPC is a generic scheme and satisfies the existing consistency schemes as follows: The source waits to update the data until receives INVREP_ACK from all CHs or up to minimum update delay at every time. Hence it satisfies the SC. When TTL decreased lower than TTL_{th}, CH utilizes the pull based scheme. If TTL is large, the data source uses push based INVREP_ACK scheme. The APPC addresses the issue in consistency maintenance cost with respect to time. If required data is available in neighbor clusters then user need not to send the query to remote source. Thus cost is decreased to access the required data. Thus it ensures the consistency levels SC, WC and DC.

Algorithm.3 APPC on a Caching node:

```
while (cache node receiving a query from neighbor)
{
if (TTL> 0) then
 Serve the query with the cache copy
else if (TTL < 0)&&(TTL< TTL_th) then
 Send a RENEW request to CH to renew the TTL from the source node
end if
if (cache copy not updated) then
 update the new TTL for cache copy
else if ( cache copy has updated) then
 update cache copy with new TTL
 serve the cache query with out delay
end if
}
end while
```

Analysis:

Push: The source maintains about its caching nodes information with their respective CHs. Whenever source wants to update the data D at time t_u, it

broadcasts the INV_REP to all CHs through source node's home CH in the network. The neighbor CHs replies the INVREP_ACK to the source in reverse path without delay. Also neighbor CHs send the intimation to caching nodes about INV_REP. Hence caching nodes have not serve the cache copies to neighbors in that period. After received INVREP_ACK from all CHs, then source initiates to update D and pushes to the home CH. Then home CH transmits the updated data into neighbor caching nodes via their respective CHs. If source does not receive INVREP_ACK from any CH, then it waits up to its tolerable minimum update delay. This approach reduces the overhead, energy and bandwidth. When MN moves from one cluster C_i to another cluster C_j it informs to the new CH_j and leave message to CH_i. Thus the source push the update with new TTL to home and neighbor CHs instead of the entire caching nodes in the network. Hence it reduces the latency.

Pull: The individual caching nodes uses pull algorithm to ensure the validity of cache copy. Some of the cache copies have most frequently accessed and remaining less frequently accessed by neighbors. Based on these constraints, the cache copies TTL is renewed from the source by their home CH. The CH maintains the TTL_{th} for cache copies. When cache copy's TTL reduced to less than TTL_{th}, CH sends the request to the source to renew TTL with query access rate. TTL value set by the source based on query access rate and forwards to the CH. Then caching node renews TTL from the CH. If $D_{query\_access\_rate}$ is the interval between successive query of D and β (value is between 0 and 1 is weighting factor for the recent and past queries), then the TTL value is renewed as:

$$TTL = \beta \times TTL + (1-\beta) \times D_{query\_access\_rate}$$

The TTL is renewed based on their query access rate. Hence, TTL is not updated for the less frequently accessed data regularly. Thus it saves the overhead, bandwidth. The caching node is serving the data to neighbors query without interruption. Thus, data access latency is reduced.

Maintaining consistency: The data consistency must maintain between the source and caching nodes.

Case 1: When TTL of cache copy is decreased less than TTL_{th}, the CH send renew query to the source. At the moment, the TTL to be renewed data is updated between past and current renew requests. The data have also updated multiple times among past and current updates, but these intervals take more than δ time. Hence, source reduces the new TTL value by multiplicative factor m and also sends the updated data to caching node:

$$TTL = TTL \text{ x } m, \; 0 < m < 1$$

Because, when TTL is minimum then caching node renews frequently. Hence it reduces the staleness by renew TTL. Thus it increases the data consistency and reduces the latency.

Case 2: Sometimes the source data is not updated between past and current TTL renewal. At the moment, source only renews TTL based on linear model:

$$TTL = TTL \text{ x } (1+ f), \; f \text{ is linear factor } 0 < f < 1$$

The TTL for least frequently updated data is increased by linear factor. Because, these cache copies does not accessed frequently by neighbors. Thus it reduces overhead, bandwidth and energy consumption.

Case 3: The source updates the data before cache copy's TTL expires. In this approach, source node sends INV_REP to the CHs before push the updated data. Hence, CHs reply the INVREP_ACK to the source instead of all caching nodes in entire network. It reduces the delay to receive INVREP_ACK. To avoid the stale hits, updated data is send to the caching nodes. Thus it increases the consistency among source and cache copy.

Case 4: When the source does not receive INVREP_ACK from all CHs, it waits for t_m. But, in proposed approach mostly CHs are not expired. The new CH is selected before current CH expires and all information gives up to new CH. This process takes only minimum time. Hence source just waits for t_m only at CH reelection time. The new CH reelected rarely. Therefore source is not waiting to update the data forever. Thus it reduces the delay. The time specifications are summarized in Table 1. In data push propagation, when TTL is expired among data update time t_u and minimum update delay t_m as in Fig. 2. Here, the source sends updated data with new TTL after received INVREP_ACK from all CHs.

Table 1: Time specification

Notation	Description
t_0	Initial time
t_u	Data update time
t_m	Minimum update time delay
t_δ	Acceptable time deviation among source and cache copy
t_r	TTL renewal request time

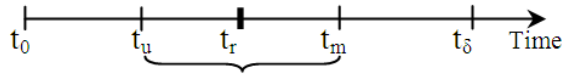

Fig. 2: TTL renew request t_r among t_u and t_m

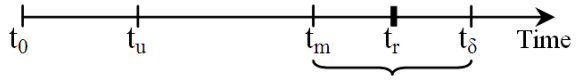

Fig. 3: TTL renew request t_r among t_m and t_δ

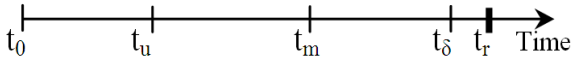

Fig. 4: TTL renew request t_r after t_δ

Hence the stale hit is not occurs due to the range (t_u, t_m) is tolerable minimum update delay:

$$t_r < t_u + t_m$$

When TTL expires among update delay t_m and acceptable deviation t_δ as in Fig. 3, stale hits have not occurred. Because, the range (t_m, t_δ) is acceptable duration between the source and cache copy:

$$t_r < t_m + t_\delta$$

The Fig. 4 shows that TTL expires after the acceptable deviation. At this moment, CH requests the source to renew the TTL once it decreases less than TTL_{th}. Hence, it renews the TTL priorly for its cache copy. Thus a stale hit is not occurs:

$$t_r > t_u + t_m + t_\delta$$

MATERIALS AND METHODS

The simulation for the proposed approach was carried out using Network Simulator Version-2 (NS2) with channel capacity is 2 Mbps. The Distributed Coordination Function (DCF) of IEEE 802.11 for wireless LANs used as the MAC layer protocol. It has the functionality to notify the network layer about link disconnection.

In this simulation, MNs make mobility in a 1000×1000 m area for 100 sec simulation time and assume each MN moves independently. The transmission range of MNs is 250 m. The network size is varied as 20, 40, 60, 80, 100 and 120 nodes and the speed of the mobile node is varied as 2, 5, 7, 10, 12 and 15 m sec^{-1}. Assume that data query and update process is based on Poisson process. The simulated traffic is Constant Bit Rate (CBR).

The simulation settings and parameters are summarized in Table 2.

Table 2: Simulation settings

Parameters	Values
No. of nodes	20,40,60,80,100,120
Area size	1000´1000
Radio range	250 m
Simulation time	100 sec
Cache size	1000 KB
Mobility model	Random way point
Speed	2, 5, 7, 10, 12, 15 m/sec
Data size	500 bytes
Avg. update interval	20s
Bandwidth	2 Mbps
TTL range	10-18 sec
TTL threshold	4 sec
Avg. query interval	5 sec

Performance metrics: The proposed Cluster Based Data Consistency (CBDC) associates with Adaptive Push and Pull Algorithms for Clusters (APPC) approach is compared with Flexible Cache Consistency (FCPP) Maintenace (Huang *et al*., 2010a) and Cluster Based Cooperative Caching Technique (CBCCT) in Mobile Ad Hoc Networks (Kuppusamy *et al*., 2012) schemes.

Average query latency: The average latency is the average latency between user sending the query to the source and receiving the reply from source.

Success ratio: The ratio of total number of queries sent to the source and total number of packets received successfully.

Control overhead: The control overhead is defined as the total number of control packets normalized by the total number of received data packets.

RESULTS

Based on nodes: The first simulation scenario was constructed by varying the number of nodes as 20, 40, 60, 80 100 and 120 with mobile speed as 5 m sec^{-1}.

When the nodes have increased in the network, caching nodes also increased. Due to additional caching nodes in clusters, the query latency and overhead have increased slightly, overall packet delivery success ratio is reduced.

The Fig. 5 shows that the proposed approach, APPC has less query latency than the existing approaches. Because CHs shares cache copy and maintain consistency of cache copy with source. In addition Fig. 6 shows that proposed APPC achieves more success ratio when compared with the existing FCPP and CBCCT. The Fig. 7 shows that APPC outperforms the existing approaches in terms of control overhead. Since the query latency and overhead have increased, the overall success ratio is decreased when the number of caching nodes is increased.

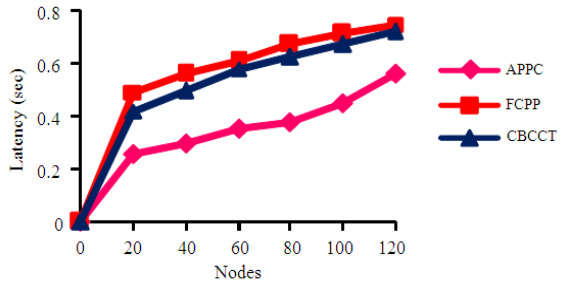

Fig. 5: Nodes Vs latency

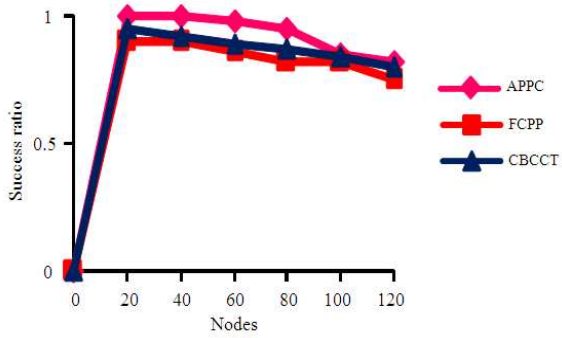

Fig. 6: Nodes Vs packet success ratio

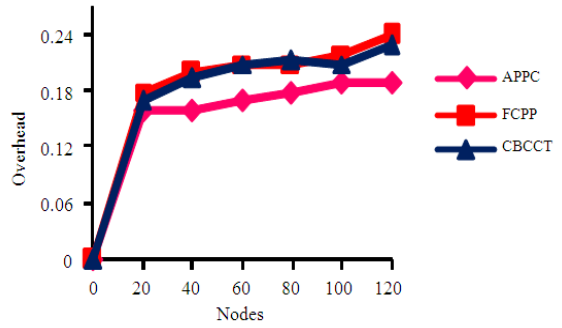

Fig. 7: Nodes Vs overhead

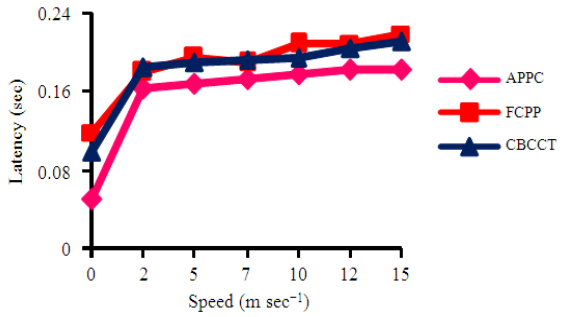

Fig. 8: Speed Vs latency

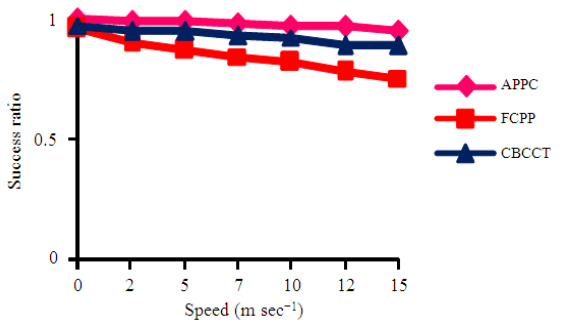

Fig. 9: Speed Vs packet success ratio

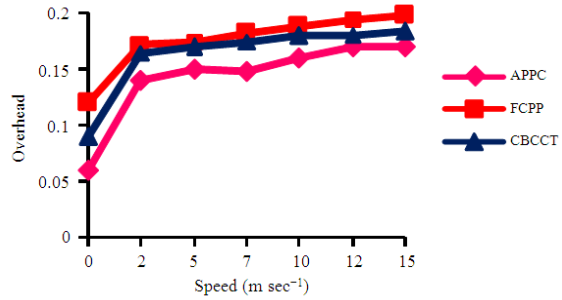

Fig. 10: Speed Vs overhead

Based on speed: This second scenario was constructed by varying the speed of the MNs as 2, 5, 7, 10, 12 and 15 m sec^{-1} with 50 nodes.

The Fig. 8 shows that the proposed APPC protocol has less query latency. Since MNs always have communication with any one of CH. Hence CH accomplishes the neighbor, home cluster MNs data requirements. Figure 9 and 10 shown that proposed APPC achieves more success ratio and less control overhead than the existing FCPP and CBCCT schemes.

DISCUSSION

The simulation shown that the proposed Cluster Based Data Consistency associate with APPC approach reduces latency 2.4%, 2% and overhead 3.5%, 3% and increases packet success ratio 9.2%, 5.5% than FCPP and CBCCT respectively with respect to increasing number of nodes. Also APPC reduces latency 2.2%, 1.7% and overhead 2.3%, 1.5% and increases packet delivery success ratio 9.3%, 3.6% than FCPP and CBCCT respectively with respect to mobility speed. The results proved that the proposed approach APPC provides better performance than the existing approaches FCPP, CBCCT.

CONCLUSION

This study presents a consistency maintenance scheme Cluster Based Data Consistency (CBDC) for cooperative caching over MANET. CBDC improves the data accessibility by reducing the latency, overhead. Also it reduces the energy consumption of MNs by partitioning the huge network into clusters. The CHs shared their information with neighbors to improve the performance. Thus the cooperative caching improves the data availability in MANET. Also Adaptive Push and Pull Algorithms for Cluster proposed to improve the data consistency among the source and caching MNs. The combination of push and pull algorithm for clusters improves the data consistency among source and cache copies by associate with TTL values. These proposed algorithms have reduced the overhead and latency, energy consumption by using the CHs in the clusters.

REFERENCES

Artail, H., H. Safa, K. Mershad, Z. Abou-Atme and N. Sulieman, 2008. COACS: A Cooperative and Adaptive Caching System for MANETs. IEEE Trans. Mobile Comput., 7: 961-977. DOI: 10.1109/TMC.2008.18

Cao, G., 2002. On improving the performance of cache invalidation in mobile environments. Mobile Netw. Appli., 7: 291-303. DOI: 10.1023/A:1015463328335

Cao, G., L. Yin and C.R. Das, 2004. Cooperative cache-based data access in ad hoc networks. Computer, 37: 32-39. DOI: 10.1109/MC.2004.1266293

Cao, J., Y. Zhang G. Cao and L. Xie, 2007. Data consistency for cooperative caching in mobile environments. Computer, 40: 60-66. DOI: 10.1109/MC.2007.123

Cao, J., Y. Zhang, L. Xie and G. Cao, 2005. Consistency of cooperative caching in mobile peer-to-peer systems over MANET. Proceedings of the 25th International Conference Distributed Computer System, Jun. 6-10, IEEE Xplore Press, pp: 573-579. DOI: 10.1109/ICDCSW.2005.53

Duvvuri, V., P. Shenoy and R. Tewari, 2003. Adaptive leases: A strong consistency mechanism for the World Wide Web. IEEE Trans. Knowl. Data Eng., 15: 1266-1276. DOI: 10.1109/TKDE.2003.1232277

Feeney, L.M., 2001. An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks. J. Mobile Netw. Appli., 6: 239-249.

Huang, Y., J. Cao and B. Jin, 2006. A predictive approach to achieving consistency in cooperative caching in MANET. Proceedings of the 1st International Conference on Scalable Information Systems, (SIS' 06), ACM Press, USA. DOI: 10.1145/1146847.1146898

Huang, Y., J. Cao, B. Jin, X. Tao and J. Lu *et al*., 2010b. Flexible cache consistency maintenance over wireless ad hoc networks. IEEE Trans. Parall. Distributed Syst., 21: 1150-1161. DOI: 10.1109/TPDS.2009.168

Huang, Y., J. Cao, B. Jin, X. Tao and J. Lu, 2010a. Cooperative cache consistency maintenance for pervasive internet access. Wirel. Commun. Mobile Comput., 10: 436-450. DOI: 10.1002/wcm.819

Imielinski, T. and D. Barbara, 1994. Sleepers and workaholics: Caching strategies in mobile environments. ACM Sigmod, 23: 1-12.

Jing, J., A. Elmagarmid, A. Helal and R. Alonso, 1997. Bit-Sequences: An adaptive cache invalidation method in mobile client/server environments. Mobile Netw. Appli., 2: 115-127. DOI: 10.1023/A:1013616213333

Kuppusamy, P., K. Thirunavukkarasu and B. Kalaavathi, 2012. Cluster based cooperative caching technique in mobile ad hoc networks. Eur. J. Sci. Res., 69: 337-349.

Li, W., E. Chan, D. Chen and S. Lu, 2009. Maintaining probabilistic consistency for frequently offline devices in mobile ad hoc networks. Proceedings of the 29th IEEE International Conference on Distributed Computing Systems, Jun. 22-26, IEEE Xplore Press, Montreal, QC, pp: 215-222. DOI: 10.1109/ICDCS.2009.23

Li, W., E. Chan, D. Chen, Y. Wang and D. Chen, 2007. Cache invalidation strategies for mobile ad hoc networks. Proceeding of the International Conference on Parallel, Sept. 10-14, IEEE Xplore Press, Xi'an, pp: 57-57. DOI: 10.1109/ICPP.2007.22

Xie, G., Z. Li, J. Chen, Y. Wei and V. Issarny *et al*., 2007. DTCS: A dynamic tree-based consistency scheme of cooperative caching in mobile ad hoc networks. Proceedings of the 3rd IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, Oct. 8-10, White Plains, New York, USA., pp: 48-48. DOI: 10.1109/WIMOB.2007.28