Caching Strategies Based on Information Density Estimation in Wireless Ad Hoc Networks

[1]Mr. B. SAIBANNA [2]Mr.T. MANOHAR [3]Dr. M.ARYA BHANU
[1]M.Tech, Lords Institute of Engineering and Technology, Himayath Sagar, RR dist, AP-INDIA, E-mail:bijlybsaibanna@gmail.com
[2]Assistant Professor, Lords Institute of Engineering and Technology
E-mail: telugumanohar@gmail.com
[3]HOD, Lords Institute of Engineering and Technology

ABSTRACT*- Data caching strategy for ad hoc networks whose nodes exchange information items in a peer-to-peer fashion. Data caching is a completely distributed scheme where each node, upon getting requested information, decide the cache drop time of the information or which content to replace to make room for the newly arrived information. These assessments are made depending on the perceived "presence" of the content in the nodes proximity, whose evaluation does not cause any additional overhead to the information sharing system. We develop a strategy where nodes, self-sufficient of each other, decide whether to cache some content and for how long. In the casing of small-sized caches, we endeavor to design a content replacement strategy that allows nodes to successfully store newly received information while maintaining the good performance of the content distribution system. Under both circumstances, each node takes decisions according to its perception of what nearby users may store in their caches and with the aim of differentiating its own cache content from the other nodes'. The result is the construction of content diversity within the nodes neighborhood so that a requesting user likely finds the desired information nearby. We suggest our caching algorithms in different ad hoc network scenarios and compare them with other caching schemes, showing that our result succeeds in creating the desired content diversity, thus important to a resource-efficient information access.*

1. INTRODUCTION

Ad hoc networks are multi hop wireless networks of small computing devices with wireless interfaces. The computing devices could be predictable computers (for example, PDA, PC or laptop) or backbone routing platforms or even embedded processors such as sensor nodes. The trouble of optimal placement of caches to reduce overall cost of accessing data is motivated by the following two defining characteristics of ad hoc networks. Initial, the ad hoc networks are multi hop networks without a central base station. Thus, isolated access of information typically occurs via multi hop routing, which can really benefit from caching to reduce access latency. Next, the network is generally resource constrained in terms of channel bandwidth or battery power in the nodes. Caching helps in falling communication, this results in investments in bandwidth, as well as battery energy. The trouble of cache placement is particularly challenging when each network node has a limited memory to cache data items.

In this paper, our focus is on developing efficient caching techniques in ad hoc networks with memory limitations. Study into data storage, access, and dissemination methods in ad hoc networks is not new. In exacting, these mechanisms have been investigated in connection with sensor networking peer-to-peer networks mesh networks world wide Web and even more general ad hoc networks. Though, the presented approaches have so far been somewhat "ad hoc" and empirically based, without any sturdy analytical foundation. In difference, the theory literature abounds in analytical studies into the optimality properties of caching and replica allocation problems. However, disseminated implementations of these techniques and their performances in complex network settings have not been investigated. It is even uncertain whether these techniques are amenable to efficient distributed implementations.

Our goal in this paper is to develop an approach that is both analytically tractable with a provable performance bound in a centralized setting and is also amenable to a natural distributed implementation. In our network form, there are multiple data items; each data item has a server, and a set of clients that desire to access the data item at a given frequency. Each node cautiously chooses data items to cache in its limited memory to minimize the overall access cost. Essentially, in this piece, we develop efficient strategies to select data items to cache at each node. In particular, we grow two algorithms—a centralized approximation algorithm, which delivers a 4-approximation (2-approximation for uniform size data items) answer, and a localized distributed algorithm, which is based on the estimate algorithm and can handle mobility of nodes and dynamic traffic conditions. Using simulations, we illustrate that the distributed algorithm performs very close to the approximation algorithm. Lastly, we show through extensive experiments on ns2 that our proposed distributed algorithm performs much better than a prior approach over a broad range of parameter values. Ours is the first work to there a distributed implementation based on an approximation algorithm for the general problem of cache placement of multiple data items under memory constraint.

Data caching strategy for ad hoc networks whose nodes exchange information items in a peer-to-peer fashion. Data caching is a fully distributed method where each node, upon getting requested information, determines the cache fall time of the information or which content to replace to make room for the newly arrived information. These choices are made depending on the perceived “presence” of the content in the nodes proximity, whose evaluation does not cause any additional overhead to the information sharing system.

We devise a strategy where nodes, self-sufficient of each other, decide whether to cache some content and for how long. In the casing of small-sized caches, we aspire to design a content replacement strategy that allows nodes to successfully store newly received information while maintaining the good performance of the content distribution system. Under both conditions, every node takes decisions according to its perception of what nearby users may store in their caches and with the aim of differentiating its own cache content from the other nodes’. The answer is the creation of content diversity within the nodes neighborhood so that a requesting user likely finds the desired information nearby. We reproduce our caching algorithms in different ad hoc network scenarios and compare them with other caching schemes, showing that our clarification succeeds in creating the desired content diversity, thus important to a resource-efficient information access.

2. RELATED WORK

Hamlet is fully distributed caching wireless ad hoc networks whose nodes exchange information item in a peer to peer fashion. In particular we address a mobile ad hoc network whose nodes might be resource-constrained devices, pedestrian users, or vehicles on city roads. Each node runs a request to request and possibly cache desired information items. Nodes in the network retrieve information items from other users that temporarily cache the requested items or from one or more gate way nodes, which can store content or quickly fetch it from the internet.

We propose, called Hamlet, aims at creating content diversity within the node neighborhood so that users likely find a copy of the different information items nearby (Regardless of the content popularity level) and avoid flooding the network with query messages. Although a similar concept has been put forward in the novelty in our proposal resides in the probabilistic estimate, run by each node, of the information presence (i.e., of the cached content) in the node proximity. The estimate is performed in a cross-layer fashion by overhearing content query and information reply messages due to the broadcast nature of the wireless channel. By leveraging such a local estimate, nodes autonomously decide what information to keep and for how long, resulting in a distributed scheme that does not require additional control messages.

The Hamlet approach applies to the following cases.

• Information presence estimation. In this case we define the reach range of a generic node that can receive a query generated by node n itself. As an example, in an ideal setting in which all nodes have the same radio range, the reach range is given by the product of TTL and the node radio range. Next we denote by f the frequency at which every node estimate the presence of each information item within the reach range, and we define as 1/f the duration of each estimation step (also called time step hereafter). The generic node n uses the information captured within its reach range, during the estimation step j, to compute the following quantities:1) a provider counter by using application-layer data and 2) a transit counter by using data that are collected through channel overhearing in a cross-layer fashion. These counters are defined as follows:

• Provider counter dic(n, j). This quantity accounts for the presence of new copies of information i’s chunks c, delivered by n to querying nodes within its range during step j. Node n updates this measure every time it acts as a provider node.(e.g., like node P in the higher plot of figure 2)

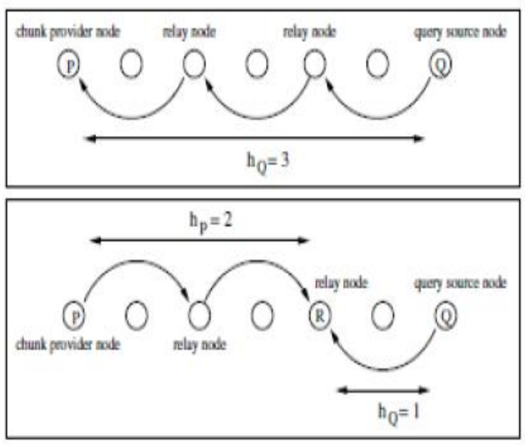


Fig. 1. Q and P denote, respectively, a node issuing a query and a node providing the requested content. Node R in the lower plot is a relay node, overhearing the exchanged messages. The plots represent: case $a)$ (upper plot) h_Q value for the provider node P, and case $b)$ (lower plot) h_Q and h_P values for relay node R, with respect to the query source Q and the provider P

• Large-sized caches. In this case, nodes can potentially amass a large portion (i.e., up to 50%) of the available information items. Reduced memory usage is a desirable (if not required) condition, as the same memory may be shared by different services and applications that run at nodes. In such a scenario, a caching decision consists of computing for how long a given content should be stored by a node that has previously requested it, with the goal of minimizing the memory usage without affecting the overall information retrieval performance;

• Small-sized caches. In this case, nodes have a committed but limited amount of memory where to store a small percentage (i.e., up to 10%) of the data that they retrieve. The caching decision translates into a cache replacement strategy that selects the information items to be dropped among the information items just received and the information items that already fill up the dedicated memory. We evaluate the performance of Hamlet in different mobile network scenarios, where nodes communicate through ad hoc connectivity. The results show that our answer ensures a high query resolution ratio while maintaining the traffic load very low, even for scarcely popular content, and consistently beside different network connectivity and mobility scenarios.

3. EVALUATION WITH LARGE-SIZED CACHES

We first compare Hamlet's performance to the results obtained with a deterministic caching strategy, called DetCache, which simply drops cached chunks after a fixed amount of time.

A. Benchmarking Hamlet

We set the deterministic caching time in DetCache to 40 s, and we couple DetCache and Hamlet with both the mitigated flooding and Eureka techniques for query propagation. We are interested in the following two fundamental metrics: 1) the ratio of queries that were successfully solved by the system and 2) the amount of query traffic that was generated. if queries hit upon the sought information in one or two hops, then the query traffic is obviously low. Thus, additional metrics that are related to cache occupancy and information cache drop time must be coupled with the aforementioned metrics.

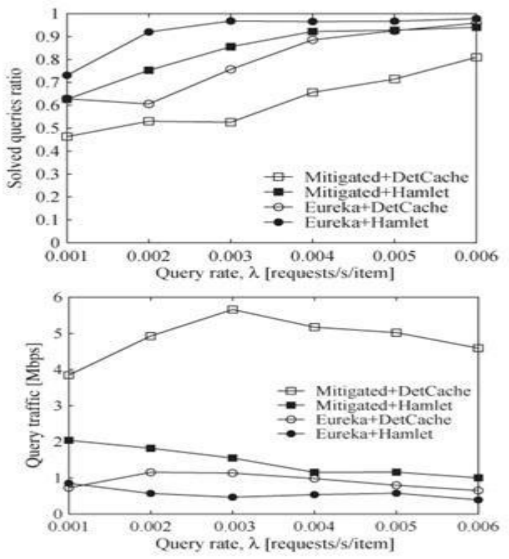


Fig. 2. City: solved-queries ratio (top) and query traffic (bottom) obtained with different schemes versus content request rate.

Fig.2 shows the solved-queries ratio (top plot) and the amount of query traffic (bottom plot) as λ varies in the City scenario. When DetCache is used, the higher the query rate, the larger the number of nodes that cache an information item. The positive effect of the caching decisions can also be observed in Fig. 5 in terms of the reduced overhead and latency in solving queries. Finally, we may wonder how well Hamlet performs with respect to DetCache when the cache time employed by the latter approach is set to a value other than 40 s. Fig. 2 refers to the Mall scenario.

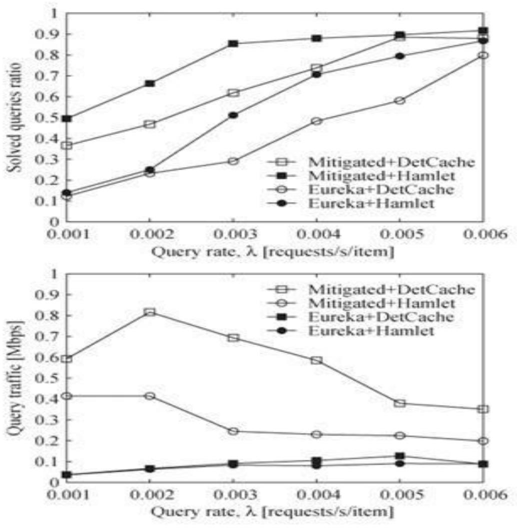


Fig. 3. Mall: Solved-queries ratio (top) and query traffic (bottom) with different schemes versus content request rate.

The poor performance of Eureka in this case is due to the lack of information items over large areas of the Mall scenario, resulting in queries not being forwarded and, thus, remaining unsolved With regard to the caching occupancy, because Hamlet leads to results that are comparable with the results obtained with DetCache (see Table I, Mall scenario), it can be asserted that the performance gain achieved through Hamlet is due to the more uniform con- tent distribution across node caches.

B. Information Survival

We say that, at a given time instant, a particular item has survived if each of its chunks is cached by at least one node in the network. These values are very close to the DetCache caching time of 40 s, showing that Hamlet improves information survival by better distributing content in the network and not by simply caching them for longer periods of time.

4. EVALUATION WITH SMALL-SIZED CACHES

The caching dynamics of the different in- formation items become strongly intertwined. Indeed, caching an item often implies discarding different previously stored content, and as a consequence, the availability of one item in the proximity of a node may imply the absence of another item in the same area. In HybridCache, a node that requests an item always caches the received data. Instead, a node on the data path caches the information if its size is small; otherwise, it caches the data path, provided that the content copy is not very far away. We set the HybridCache parameters so that the following two conditions are satisfied: 1) The size of the data never results in data path caching but always in information caching, and 2) mitigated flooding is always employed for query forwarding.

A. Benchmarking Hamlet

Fig. 7 presents the solved- queries ratio and the overall query traffic versus the information set size. We observe that Hamlet reacts better to the growth of the number of items than HybridCache, without incurring any penalty in terms of network load, as shown by the comparable query traffic generated by the two schemes.

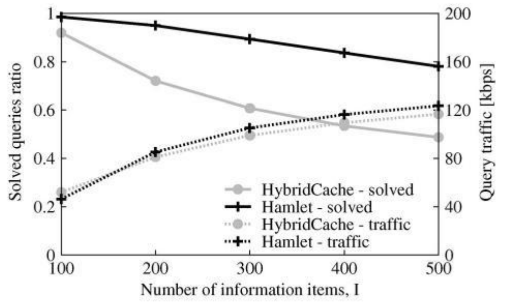


Fig.4. Static memory-constrained nodes: Solved-queries ratio and query traffic as the information set size varies, with HybridCache and Hamlet.

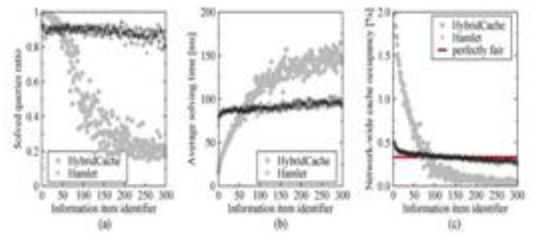


Fig. 5. Static memory-constrained nodes. (a) Query-solving ratio, (b) time, and (c) average networkwide cache occupancy for each item when using HybridCache and Hamlet, with I = 300. In (c), the red horizontal line represents perfect fairness in cache occupancy among different items.

Observing the performance of Hamlet and HybridCache on a per-item basis allows a deeper understanding of the results. In Fig. 5(a), we show the solving ratio of the queries for each item when I = 300. Along the x-axis, items are ordered in decreasing order of popularity, with item 1 representing the most sought-after information and item 300 the least requested information. Unlike Hamlet, HybridCache yields extremely skewed query solving ratios for the different content; a similar observation also applies to the time needed to solve queries, as shown in Fig. 5(b). The explanation for such behavior lies in the distribution of information in the network. Fig. 8(c) depicts the average percentage of memory used to cache a given item, aggregated over all network nodes. HybridCache fosters the storage of popular content, whereas it disregards content that is less requested, even if it represents two thirds of the whole information set. This case happens with HybridCache, as proven by the spatial distribution of the 100th, 200th, and 300th substance, as shown in Fig. 6(a). Conversely, the spatial distribution achieved by Hamlet, as shown in Fig. 6(b), is more uniform, leading to a faster more likely resolution of queries.

The z-axis in the plots shows the mean content completeness in each spatial slot, with a value of 1, meaning that the entire content can be found in the same spatial slot. (a) HybridCache. (b) Hamlet. We now compare the performance of HybridCache and Ham- let in the scenario with memory-constrained mobile nodes. We test the two schemes when I = 300 and for an average node speed vm equal to 1 and 15 m/s.

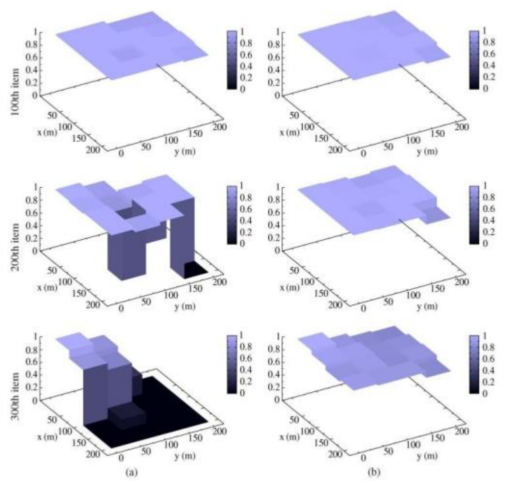


Fig.6. Static memory-constrained nodes: Spatial distribution of the 100th, 200th, and 300th items, averaged over time, for Zipf distribution exponents under HybridCache and Hamlet, with I = 300.

The solved-queries ratio recorded with HybridCache and Hamlet on a per-item basis are shown in Fig. 7. Comparing the left and right plots, we note that the node mobility, even at high speed, does not seem to significantly affect the results due to the high network connectivity level. The spatial redistribution of content induced by node movements negatively affects the accuracy of Hamlet's estimation process, which explains the slight reduction in the solved query ratio at 15 m/s. That same movement favors HybridCache, at least at low speed, because it allows unpopular information to reach areas that are far from the gateway. However, the difference between the two schemes is evident, with Hamlet solving an average of 20% requests more than

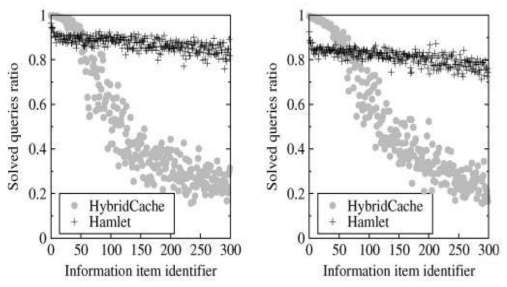


Fig.7. Memory-constrained mobile nodes: Query-solving ratio for each information item when using HybridCache and Hamlet, with I = 300. The plots refer to vm that is equal to 1 m/s (left) and 15 m/s (right).

B. Impact of the ZIPF Distribution Skew ness

We study the impact of the Zipf distribution exponent on the performance of the cache replacement strategies. We remember that an exponent that is equal to zero implies perfect homogeneity, i.e., Zipf distribution that degenerates into a uniform distribution, whereas the difference in popularity among content becomes much more unbalanced as the exponent grows. . The choice of this setting is mandated by the fact that, in the presence of hundreds of different items, unbalanced popularity distributions (i.e., exponents higher than 0.5) lead to very low λi for the 100 or so least popular items, thus making requests for such content extremely rare.

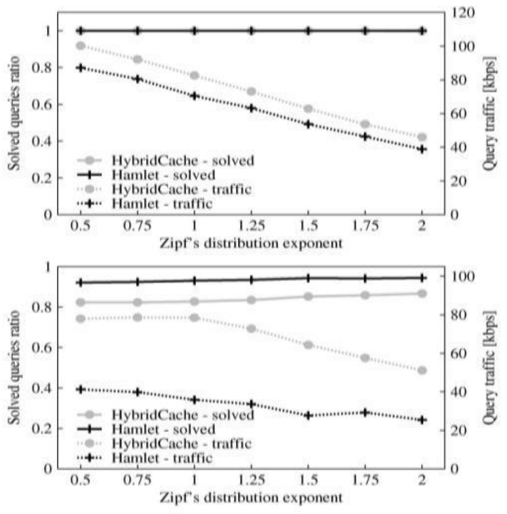
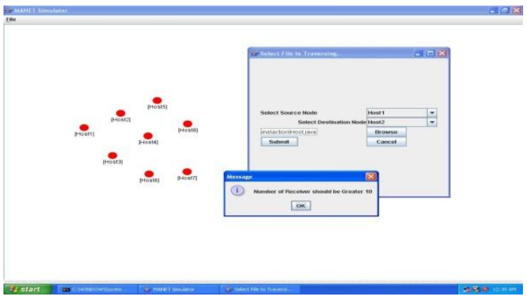


Fig.8. Memory-constrained static (top) and mobile (bottom) nodes: Solved- queries ratio and query traffic as the Zipf distribution exponent varies when using HybridCache and Hamlet, with I = 10.

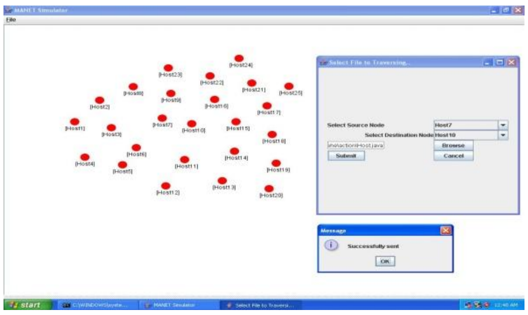
Fig.8 depicts the evolution of the solved-queries ratio and the query traffic as the Zipf exponent ranges vary. By comparing the two plots, we note that the presence of mobility (vm = 1 m/s) leads to a higher number of unsolved requests and in a larger amount of traffic generated within the network under HybridCache, because queries propagate far from the source without finding the desired item. On the one hand, higher values of the exponent lead to more unbalanced query rates, with very few items that are extremely popular and a long tail of seldom-accessed data. On the other, when the Zipf exponent is small, the distribution of queries is more balanced, with information more evenly distributed in the network.

5. RESULTS



Select Source Node Host1
Select Destination Node Host2
Message
Number of Receiver should be Greater 10

Fig. 9 Minimum number of nodes



Select Source Node Host7
Select Destination Node Host10
Message
Successfully sent

Fig. 10 Successful File Transfer

6. CONCLUSION

We have developed a paradigm of data caching techniques to support effective data access in ad hoc networks. In exacting, we have considered memory capacity constraint of the network nodes. Data caching strategy for ad hoc networks whose nodes exchange information items in a peer-to-peer fashion. Data caching is a completely distributed scheme where every node, upon receiving requested information, conclude the cache drop time of the information or which content to replace for the newly arrived information. We have developed efficient data caching algorithms to determine near optimal cache placements to maximize reduction in overall access cost. Reduction in access cost guides to communication cost savings and hence, improved bandwidth usage and energy savings. However, our simulations over a wide range of network and application parameters show that the performance of the caching algorithms. Presents a distributed implementation based on an approximation algorithm for the problem of cache placement of multiple data items under memory constraint. The result is the creation of content diversity within the nodes neighborhood so that a requesting user likely finds the desired information nearby. We replicate our caching algorithms in different ad hoc network scenarios and compare them with other caching schemes, showing that our solution succeeds in creating the desired content diversity, thus leading to a resource-efficient information access.

7. REFERENCES

[1] Marco Fiore, Member, IEEE, Claudio Casetti, Member, IEEE, and Carla-Fabiana Chiasserini, Senior Member, IEEE

[2] G. Cao, L. Yin, and C. R. Das, “Cooperative cache-based data access in ad hoc networks,” Computer, vol. 37, no. 2, pp. 32–39, Feb. 2004.

[3] C.-Y. Chow, H. V. Leong, and A. T. S. Chan, “GroCoca: Group-based peer-to-peer cooperative caching in mobile environment,” IEEE J. Sel. Areas Commun., vol. 25, no. 1, pp. 179–191, Jan. 2007.

[4] T. Hara, “Cooperative caching by mobile clients in push-based information systems,” in Proc. CIKM, 2002, pp. 186–193.

[5] L. Yin and G. Cao, “Supporting cooperative caching in ad hoc networks,” IEEE Trans. Mobile Comput., vol. 5, no. 1, pp. 77–89, Jan. 2006.

[6] N. Dimokas, D. Katsaros, and Y. Manolopoulos, “Cooperative caching in wireless multimedia sensor networks,” ACM Mobile Netw. Appl., vol. 13, no. 3/4, pp. 337–356, Aug. 2008.

[7] Y. Du, S. K. S. Gupta, and G. Varsamopoulos, “Improving on-demand data access efficiency in MANETs with cooperative caching,” Ad Hoc Netw., vol. 7, no. 3, pp. 579–598, May 2009.

[8] W. Li, E. Chan, and D. Chen, “Energy-efficient cache replacement policies for cooperative caching in mobile ad hoc network,” in Proc. IEEE WCNC, Kowloon, Hong Kong, Mar. 2007, pp. 3347–3352.

[9] M. K. Denko and J. Tian, “Cross-layer design for cooperative caching in mobile ad hoc networks,” in Proc. IEEE CCNC, Las Vegas, NV, Jan. 2008, pp. 375–380.

[10] B.-J. Ko and D. Rubenstein, “Distributed self-stabilizing placement of replicated resources in emerging networks,” IEEE/ACM Trans. Netw., vol. 13, no. 3, pp. 476–487, Jun. 2005.

[11] J. Cao, Y. Zhang, G. Cao, and L. Xie, “Data consistency for cooperative caching in mobile environments,” Computer, vol. 40, no. 4, pp. 60–66, Apr. 2007.

[12] N. Dimokas, D. Katsaros, and Y. Manolopoulos, “Cache consistency in wireless multimedia sensor networks,” Ad Hoc Netw., vol. 8, no. 2, pp. 214–240, Mar. 2010.

[13] H. Chen, Y. Xiao, and X. Shen, “Update-based cache replacement policies in wireless data access,” in Proc. BroadNets, Boston, MA, Oct. 2005, pp. 797–804.

[14] J. Xu, Q. Hu, W.-C. Lee, and D. L. Lee, “Performance evaluation of an optimal cache replacement policy for wireless data dissemination,” IEEE Trans. Knowl. Data Eng., vol. 16, no. 1, pp. 125–139, Jan. 2004.

Author's Profile

Mr.T.MANOHAR working as Assistant Professor at Lords Institute of Engineering and Technology, Himayath Sagar, RR dist, AP-INDIA.

Mr. B. SAIBANNA is currently Pursuing M. Tech in the College of Lords Institute of Engineering and Technology, Himayath Sagar, RR dist, AP-INDIA. He has received his B. Tech from Sreenidhi Institute of Science and Technology, Yamnampet, Ghatkesar, Hyderabad, AP-INDIA.