An Efficient Evolutionary and Cache Based Cooperative Communication in Wireless Sensor Network

M.Srinivasa Rao[1] and N.Sarath Babu[2]
[1] *M.Tech Student,* [2] *Asst. Professor*
[1,2]*CSE Dept., Swarnandhra Institute of Engineering and Technology, Narsapur.*

Abstract: Cooperative communication is the one of the current interesting research issue in the field of wireless sensors, various cache based approaches proposed by various authors but cache individually cannot increase the performance over MANET. Topology architectures defined for data transmission and cooperative communication in MANET, In this approach we are introducing an empirical model for cooperative communication with one of the evolutionary algorithm along with cache implementation which is proposed in previous mechanism, In this approach we consider the factors of signal strength and channel capacity for calculating the communication cost then we generates the chromosomes for data transmission between source and destination through intermediate nodes.

Index terms: Cooperative communication, Cache, MANET, Wireless Sensor Networks

I. INTRODUCTION

Mobile adhoc networks have many applications such as web conferences, tourist centers, Wireless offices etc. For increasing the accessibility of data mobiles nodes or devices should maintain cache to store various data. By increasing the data accessibility processing of the query response takes more time. So accessing data from neighbor nodes locally decreases the processing time and increases the accessibility.

In this cooperative catching plays crucial role in accessibility of data. Cooperative catching consists of multiple nodes sharing and maintenance of the cached data. In wired networks mostly use this cooperative catching to increase the performance. By using this mobile nodes can modify the route and send the data to the requested node. So this process reduces the modification of the data in wireless networks. Using authentication method user can authenticate the data is authenticated or not which means that the data received from the original source and not modified in the network channel even another data receivers are not trusted.

Identifying the data source is authenticated or not is complex task because any receiver with the shared key can copy the data and duplicate the sender. Attaching the authentication signature with the data using shared key does not work every time because of impersonation. So we adopt cryptographic techniques to generate digital signatures using private key (which is not shared in the network) for authentication. Mobiles nodes can verify the combination of the signature and data using the source public key. There are two types of cooperative catching techniques. They are cooperative catching and layered catching.

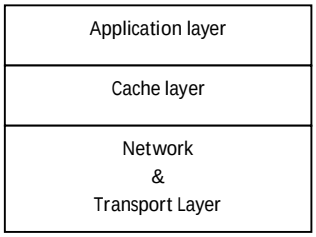
Cooperative catching functions are linked with network layer. Then the node can check every packet send to requested node. This approach has some drawbacks such as network layer mainly linked with kernel so it is very difficult to customize it.

Layered caching is having two options such as cross layer information based function and another is network layer using TCP or IP. Coming to the cross layer based functions, at the time application sends data request to routing layer. This approaches leads more complexity to routing process required to maintain the cache data as table. So it deals with fragment situations of the data not pass through the requested node. Another one if any node A request data from another node B, based on routing method the sender node know neighbor node C and sends request to neighbor node C attaching the request message. Then C receives the request and it sends the request of A to cache layer and that can check if the request serves locally or not. This sequential process continues until the request serve node A. This solution also has some drawbacks such as to server corrupted free cache data more protocols need TCP layer. So we have to move TCP Layer at every node in the network. In this process data only send to routing layer if cooperating cache is not used when request data.

Further researches focused on asymmetric cooperative caching techniques. In this we have three phases such as forwarding the request message, determining the caching node a forwarding the data reply. In the below section explained its functionality in detail. Then in section III we explained our proposed work.

II. RELATED WORK

Cooperative caching was implemented in wireless p2p networks to cache the data. It was based on asymmetric approach. In asymmetric approach to cache the data, a layered design was considered. Cooperative cache is designed as a middleware lying right below the application layer and on top of the network layer (including the transport layer). In asymmetric cooperative cache approach, the data requests are transmitted to the cache layer on every node, but the data replies are only transmitted to the cache layer at the intermediate nodes that need to cache the data. This solution not only reduces the overhead of copying data between the user space and the kernel space, it also allows data pipelines to reduce the end-to-end delay.



A Cache Node

Asymmetric cooperative caching:

In this there are three stages and those are presented below.

Requested Message Forwarding:

In this initially application generates request message and sends to cache layer. Cache layer merges the original message with destination address to reach the real destination. In this process consider the cache access the routing data and finds the next node to read the destination. This process is done by the routing protocol depend on the DSR routing protocol.

At the time of the receiving the request message it delivers that request to cache layer. Initially it checks the requested serves locally otherwise it checks whether to cache the requested data based on decision formula suggested. The decision examines its local status that is the access rate of this data request distance to the requested node and route status. If the result is to cache the requested data then its node identity will Cache List that is linked list merged in the cache layer. When the request message reaches the node has the data of the request the Cache List in the message will include entire intermediate nodes with the forwarding route which want to cache the requested

Determining the cache node:

At the time of request message reaches the destination node that has cached the requested data the cache manager verifies the Cache List and it makes the final decision on which ones in the Cache List will cache the data and it may deletes some nodes if required. Advantage of allowing the data server reconsider the caching result is that the requested node can use more parameters to clarify the purpose of caching. For example, the data centre can add update ratio as another parameter and re-evaluate the caching decision. If many intermediate nodes decide to cache the data based on the formula provided in the data server can compare the relevance of each node, and select those with highest relevance into Cache List, to avoid generating too much unnecessary cache data. Also, if the geographic location or hop distance of the intermediate node in Cache List is attached, the data server can better determine the distribution of the caching nodes, e.g. to avoid too many data replicas in one area.

Sending Data Reply:

Unlike the data request, the data reply only needs to be processed by those nodes that need to cache the data. To deliver the data only to those that will cache the data, tunnelling techniques are used. The data reply is encapsulated by the cache manager, and tunnelled only to those nodes appeared in Cache List.

The Asymmetric Cooperative Cache (ACC) is one of the approaches in various network environments. Simple-Cache is the traditional cache scheme that only caches the received data at the query node. We also compare these schemes to an Ideal Cooperative Cache (ICC) approach, which does not have processing delay at the cache layer. Further, upon receiving each packet, the cache manager makes a copy of the packet and buffers it, and then forwards the original one immediately. Thus, an intermediate node can immediately forward the packet without waiting until the whole data item is received, which can maximize the pipeline effect. It is easy to see that ICC sets up a performance upper bound that any cooperative cache scheme can achieve.

Algorithm for asymmetric approach:

1. Initialize Cache cluster heads and gateways by finding centroids for different transmission ranges.

2. Centroid for n points is calculated as follows:
X = (x1+x2+…….+xn)/n and
Y = (y1+y2+…….+yn)/n

3. Cache the status of the nodes by updating cache tables maintained by Cache cluster heads.
4. Choose source and destination nodes.
5. Find shortest path from source to destination by using DSR or AODV
6. Transmit the data if the nodes on the chosen path are all active by retrieving status from Cache cluster heads.

III. PROPOSED WORK

In our work we propose a topology, we presented as special algorithm for moderate cooperative communication between the nodes having parameters network channel capability, strength of signal and temporary memory that is cache implementation for previous accessed / transferred data for accessing. It tends to the increasing of communication cost, Hence we introduced genetic algorithm leads to the optimal solution for decreasing the communication cost. It applies the method for path selection and mutation operation between the nodes. Then mutation operation once again computes the communication cost between the source node and the destination node followed by relay nodes.

In the initialize the communication between the nodes we connect through socket programming. The node which is connected with another node communication each other at the time of data packet transfer. Each node in the communication acts as a server and accept requests from another nodes. It receives data packets from accepted nodes and vice versa.

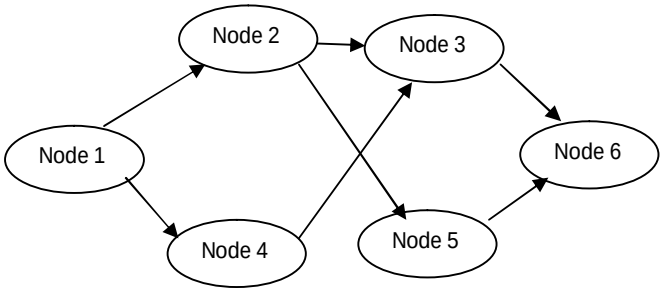


Fig -1

Proposed Approach:

Genetic algorithm is a process which uses the operators to generate off spring of the previous group of chromosomes. Here we explained about the operators present in genetic algorithm such as Selection, Crossover and Mutation.

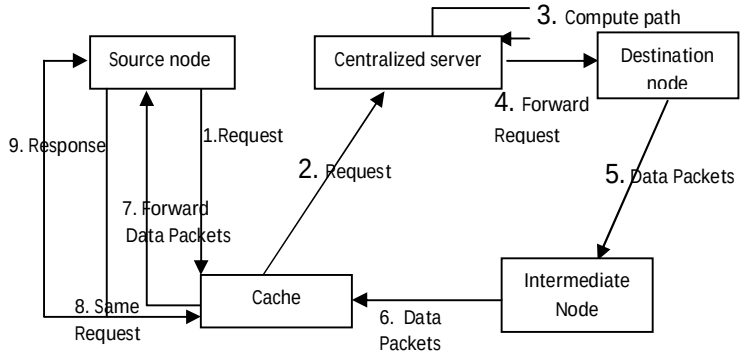
Selection: This operator selects a chromosome in existing set of chromosomes based on fitness. It copies that selected chromosome without any changes into the new chromosomes. It uses wheel selection that depends on fitness value of the chromosomes in each generation and the best fittest chromosomes more chances to get selected.

Crossover: This operator generates new chromosomes based on particular probability from two selected chromosomes. It swaps segments in chromosomes at particular position in chromosomes produces new chromosome.

Mutation: This operator generates new chromosomes by interchanging the genes in chromosomes itself.

Our Architecture:

The below architecture show our complete proposed work. The Source node sends request to the global (centralized) server using cache. If the data is not present in cache the centralized server calculates the optimal path by calculating the optimal cost and sends the data packets using the path in secure channel. If the requested data packet is available at cache there is no need to connect with centralized server otherwise it connects with server and copy data packet from the centralized server and then copy to cache.



Analysis of Optimal Communication cost:

At cooperative communication between the nodes are communicate each other with optimal path which is generated by genetic algorithm. At the time of data transfer to receiver the source node calculates communication cost and optimal path using genetic algorithm (evolutionary algorithm). Then the source node selects one of the paths from the set of optimal paths to transfer the data to destination node.

Communication cost (complexity) =Signal-strength + channel capability gets the optimal path which has the best communication cost and transfer the data packet through the path.

1. Source node chooses the destination to transfer the data.
2. If the request received by the processing method it generates the paths in architecture.
3. The Processing method calculates the path with their signal strength and channel capability.
4. Then compute the communication complexity with signal strength and network channel capacity for fitness value.
5. Select optimal communication cost and transfers the data.

Here we explain an example, Take some set of nodes A,B,C,D,E,F and if a node 'A' wants to send the data to receiver 'F' , The processing module calculates all the available paths from source to destination. Then apply the fitness value and obtains the optimal path and transfer the data over that path using the following Evolutionary approach as shown below

A→B→C→D→E→F

A→B→E→D→C→F

A→E→D→C→B→F

A→C→D→B→E→F

Then compute the fitness value based on the signal strength and channel capacity as communication cost and Obtains the optimal path which has the best communication cost and transmits the data over the path.

IV. CONCLUSION

Finally we conclude our research work with efficient cache implementation and evolutionary routing protocol based on signal strength and channel capacity for calculation of communication cost. Our primary factors give optimal performance than the traditional weight based approaches an cache improves the performance by reducing the response time of the requested node.

V. FUTURE ENHANCEMENT

We are concluding our research work with efficient routing approach for cooperative communication and cache implementation for frequently accessed information. It leads to optimal of usage of bandwidth, reduces the network traffic and improves in terms of time complexity. We can enhance our approach by reducing the time complexity issues in the split cache replacement and by implementing in our current approach.

REFERENCES

[1] *Distributed Cooperative Caching in Social Wireless Networks*. Mahmoud Taghizadeh, Kristopher Micinski, Charles Ofria, Eric Torng, and SubirBiswas

[2] *Improving On-Demand Data Access Efficiency InManets With Cooperative Caching* By Yu Du
[3] *A Survey of Web Cache Replacement Strategies*Stefan podlipnig and laszlobo¨ szo¨ rmenyi
[4] A. Chankhunthod and P. B. Danzig and C. Neerdaels and M. F. Schwartz and K.J. Worrell, *A hierarchical internet object cache," in USENIX Annual Technical Conference, 1996.*
[5] L. Fan and P. Cao and J. Almeida and A. Z. Broder, \Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, pp. 281{293, 2000.
[6] S. Iyer and A. Rowstron and P. Druschel, \Squirrel: *A decentralized peer-to-peer web cache*," in *PODC*, 2002.
[7] S. Podlipnig and L. Boszormenyi, "*A Survey of Web Cache Replacement Strategies*," ACM Computing Surveys, vol. 35, pp. 374-398, 2003.
[8] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott, "*Impact of Human Mobility on Opportunistic Forwarding Algorithms*," IEEE Trans. Mobile Computing, vol. 6, no. 6,pp. 606-620, June 2007.
[9] "*BU-Web-Client - Six Months of Web Client Traces*," http://www.cs.bu.edu/techreports/1999-011-usertrace-98.gz, 2012.
[10] A. Wolman, M. Voelker, A. Karlin, and H. Levy, "*On the Scale and Performance of Cooperative Web Caching,*" Proc. 17th ACM Symp.Operating Systems Principles, pp. 16-31, 1999.
[11] S. Dykes and K. Robbins, "*A Viability Analysis of Cooperative Proxy Caching,*" Proc. IEEE INFOCOM, 2001.
[12] M. Korupolu and M. Dahlin, "*Coordinated Placement and Replacement for Large-Scale Distributed Caches*," IEEE Trans. Knowledge and Data Eng., vol. 14, no. 6, pp. 1317-1329, Nov. 2002.