PTEA algorithm for Wireless P2P Networks in the presence of cooperative cache

Karunakar Kothapelli\textsuperscript{#1}, S Santhosh\textsuperscript{#2}

\textsuperscript{#1}computer science and Engineering, perusing M.tech in Sreenidhi institute of Science and Technology, Hyderabad, Andhra Pradesh, India.

\textsuperscript{#2}computer science and Engineering, 1Jawaharlal Nehru University assoc.prof, Sree Chaitanya College of engineering, Karimnagar, Andhra Pradesh, India.

Abstract— Improving the performance of p2p networks such as Ad Hoc networks and Mesh Networks is nontrivial. Many algorithms proposed to improve the performance of Ad Hoc networks in the presence of cooperative cache. We propose a novel dynamic algorithm to improve asymmetric cooperative cache, where 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. Novel algorithm ‘PTEA’ Which Efficiently calculates the Optimized path between nodes in network and determines the optimized node to Cache the data.

Keywords— Ad Hoc Networking, Cooperative Caching, PTEA.

1. INTRODUCTION

Ad Hoc network is an autonomous system of mobile hosts connected by wireless links, the union of which forms a communication network modelled in the form an arbitrary communication graph. A MANNET environment is characterised by energy-limited nodes, bandwidth-constrained, variable-capacity wireless links and dynamic topology, leading to frequent and unpredictable connectivity changes. The main aim of our work is to reduce the overhead of node calculation as they are energy-limited and Bandwidth Constrained.

If the nodes in the networks are able to cooperate with each to cache and share the data then power can be saved and bandwidth can be utilized properly. For example, in a battlefield, a wireless P2P network may consist of several commanding officers and a group of soldiers. Each officer has a relatively powerful data center, and the soldiers need to access the data centers to get various data such as the detailed geographic information, enemy information, and new commands. The neighbouring soldiers tend to have similar missions and thus share common interests. If one soldier has accessed a data item from the data center, it is quite possible that nearby soldiers access the same data some time later. Cooperative Caching has implemented in [2],[6],[8],[11] which allow sharing and coordination of cached data among multiple nodes. But these [2], [6], [8], [11] did not specify the efficient bandwidth utilization.

There have been many implementations of wireless ad hoc routing protocols. In [3], Royer and Perkins suggested modifications to existing kernel code to implement AODV. In [10], the authors explored several system issues regarding the design and implementation of routing protocols for ad hoc networks. Dynamic Source routing (DSR) [4] implemented by Monarch project in FreeBSD. This implementation was entirely in the kernel and made extensive modification in the kernel IP packet. However none of them looked into bandwidth utilization and route discovery in the presence of cooperative cache.

Security is extremely important for the deployment of a Mobile Ad-hoc Networks (MANET) due to its openness to attackers, the absence of an infrastructure, and the lack of centralized administration. Most research efforts have been focused on secure routing protocols [7]. Secure auto configuration and public-key distribution [1], namely the SAPKD scheme in this paper. It guarantees the uniqueness of IP address allocation. At the same time, it distributes the public key of the new node to all (or most) members in the MANET. In the ideal situation, all the nodes will receive the binding of the public key and IP address from the new node.

Although cooperative caching has been implemented by many researchers [5], [9], these implementations are in the web environment, and all these implementations are at the system level. As a result, none of them deals with on-demand nature of the ad hoc routing protocols. To realize the benefit of cooperative cache [2], intermediate nodes along the path need to check every passing by packet to see if the cached data match the data request.

We study the traffic and utilization of bandwidth between the nodes in the presence of cooperative cache. Since in the MANNET environment Nodes are free to move arbitrarily with different speeds; thus, the network topology may change randomly and at unpredictable times. So it is effective to have alternative paths from source to destination. When a data request comes to the data center then the data center selects the optimized route to the source and finds the optimized node along the path to cache the data to server further requests.
2. IMPLEMENTATION OF ’PTEA’ ALGORITHM

In this section, first we present the basic idea of three cooperative caching schemes proposed in [2], [11]: CachePath, CacheData and asymmetric cooperative caching approach [2]. Then we present our PTEA algorithm.

A. Cooperative caching schemes

Fig. 1 illustrates the Cache Path concept. Suppose node N1 requests a data item (D1) from N0. When N3 forwards D1 to N4, N5 knows that N1 has a copy of the data. Later, if N2 requests D1, N3 knows that the data source N0 is four hops away whereas N1 is only two hops away. Thus, N3 forwards the request to N1 instead of N4. Many routing algorithms such as AODV [3] and DSR [4] provide the hop count information between the source and destination. Caching the data path for each data item reduces bandwidth and power consumption because nodes can obtain the data using fewer hops.

In the CachePath, a node need to record the path information of all passing by packets rather, it only records the data path when it is closer to the caching node than the data source. If N0 forwards D1 to the destination node N1 along the path N5—N4—N3, N4 and N5 won’t cache D1 path information because they are closer to the data source than the data center.

In CacheData, the intermediate node caches the data when it finds that the data item is frequently accessed. For example, in fig.1, if both N6 and N7 request D1 through N5, N5 may think that D1 is popular and cache it locally.

Since the data replies are no need to pass through the cache layer of every node. If no intermediate node needs to cache the data, N1 sends the requested data item D1 directly to the N0 without going to cache layer of every node. If intermediate node needs to cache the data then data server determines those caching nodes and sends requested data to caching nodes.

B. Asymmetric Cooperative caching

The asymmetric cooperative caching is implemented by changing the address and next hop by increasing the hop count. The node increases the request count if the no of requests for the same data item and availability of bandwidth from the routing table and in the routes between the nodes. And calculates the availability of bandwidth based on these two factors (Traffic and Bandwidth). Through these, the data center calculates the optimal intermediate nodes to cache the data.

The algorithm has four steps

STEP 1: Forwarding the Request Packet. After a request message is generated by the node it adds the destination address Dest_addr, Data_id and next hop address so that data request can reach the destination. This can be accomplished by AODV or DSR routing algorithms.

When intermediate node receives the data request it delivers to the cache layer, the cache manager checks whether requested data item available in local cache or not if available it serves else forwards to next neighbour by changing the address and next hop by increasing the hop count. The node increases the request count if the no of request for a same data is requested.

STEP 2: Calculating the best route. The center will takes the hop count, no of requests for the same data item and availability of width from the routing table and request packet count then calculate the best route and the best node to cache the data and best node to cache the path along the discovered route. Here PTEA algorithm calculates the free slots in the links available from data center to requesting node based on the available links the data center selects that path to send the requested data item.
STEP 3: Determining the CacheData node. After a request message reach to the data center, it determines the optimized node to cache the data. While determining CachedData node it checks for no of requests for the same data item, no of links from that node to another node and the bandwidth and free slots along the link. The data center counts the no of requests for the same data which is available in the Req_count, path information and free slots from above second step then determines the CacheData node. After determining the CacheData node it finds CachePath node along the path.

STEP 4: Data Reply. The data center sends the requested data to the intermediate nodes which are selected to cache the data along the path discovered by data center. So the CacheData node saves the data in its local cache for future use and forwards the data to requester. If the CachePath node encounters the message then it saves the path information of Cache Node.

![Figure 4 System Architecture.](image)

4. CONCLUSION

In this paper we present a novel PTEA algorithm for wireless P2P networks in the presence of cooperative cache which will finds the best place to cache the data and the alternative routes to transfer the data from data center to the requesting node. The data center will effectively finds all optimized nodes to cache data for future and sends the data to those intermediate CachingNodes in effective manner using the concept of bandwidth utilisation of the network.

ACKNOWLEDGMENT

The work presented in this paper is supported by many authors so we would like to thank all the authors for their innovative work.

REFERENCES


[2] Cooperative caching in wireless P2P networks; Design, Implementation and Evolution. By Jing Zhao, Student Member, IEEE, Ping Zhang, Guohong Cao, Senior Member, IEEE, and Chita R. Das, Fellow, IEEE.


