International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622
www.ijera.com Vol. 2, Issue 2,Mar-Apr 2012, pp.903-908

Implementation Of Cd-Mac Protocol Algorithm For Reliable Communication In Wireless Ad Hoc Networks

V V K LAKSHMI[*1], Dr K VENKAT SUBBAIAH[*2]
[*1]Assistant Professor, Department of mechanical engineering, VIET Visakhapatnam, India
[*2]Proffessor, Department of Mechanical Engineering, Andhra University, Visakhapatnam, India

Abstract
In interference-rich and noisy environment, wireless communication is often impeded by unreliable communication links. Primitively there has been active research on cooperative communication that enhances the communication reliability by having a collection of radio terminals transmit signals in a cooperative way. This paper is here to endeavour a medium access control (MAC) algorithm, called Cooperative Diversity MAC (CD-MAC) that exploits the cooperative communication capability of the physical (PHY) layer to rehabilitate robustness in wireless ad hoc networks. In CD-MAC, each terminal tactically selects a partner for cooperation and lets it transmit simultaneously in consequence of this the interference from nearby terminals can be alleviated and thus enhances the network performance. For feasibility, CD-MAC is designed based on the widely adopted IEEE 802.11 MAC protocol. For accurate evaluation, this study presents and uses a realistic reception model by taking bit error rate (BER), derived from Intersil HFA3861B radio hardware, and the corresponding frame error rate (FER) into consideration. System-level simulation study shows that CD-MAC significantly outperforms the original IEEE 802.11 MAC in terms of packet delivery ratio and end-to end delay.

Keywords – Ad Hoc, MAC, Cooperative, P2P Network, cachepath.

Introduction
Wireless P2P networks, such as ad hoc network, mesh networks, and sensor networks, have received considerable attention due to their potential applications in civilian and military environments such as disaster recovery efforts, group conferences. Most of the previous researches in ad hoc networks focus on the development of dynamic routing protocols that can efficiently find routes between two Communicating nodes. Although routing is an important issue in ad hoc networks, other issues such as information (data) access are also very important we use the following example to motivate our research. In MANETs, all communications are done over wireless media, typically by radio packet through the air, without the help of wired base stations as in Fig.1. Direct communication is allowed only between adjacent nodes, so distant nodes communicate over multiple hops, and nodes must cooperate with each other to provide routing.

Fig.1: Ad Hoc network Setup

Recently, there has been active research in developing cooperative MAC algorithms such as path-centric medium access and MAC-layer packet relaying as shown in Fig.2. For example, in CoopMAC and rDCF, cooperating relay nodes are determined in a proactive manner and are used to forward frames at higher bit rates. Their objective is to deliver frames faster by using multi-rate capability, which does not necessarily enhance the communication reliability in interference-rich environment. On the other hand, cooperative communication at the PHY layer attracts a lot of researchers' attention because it directly enhances the link reliability. It refers to scenarios in which distributed radios interact with each other to jointly transmit information in wireless environments. In other words, cooperative communication exploits diversity offered by multiple users, known as multiuser or cooperative diversity.

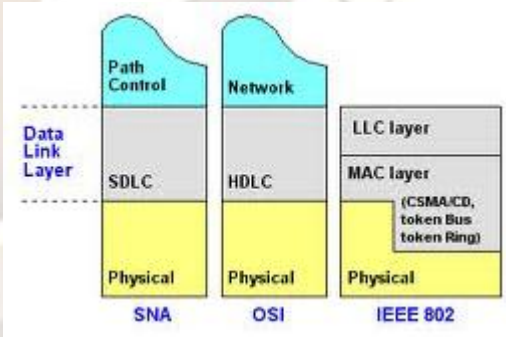

Fig.2: MAC Layer in OSI Model

It dramatically improves bit error rate (BER), resulting in a more reliable transmission and a higher throughput. It is important to note that the primary motivation of cooperative diversity in this paper is to improve the link reliability over wireless fading channels while that in previous studies is to lengthen the transmission range. Our research work focused on the design and implementation of cooperative cache in wireless P2P networks. Through real implementations, we identify important design issues and propose an asymmetric approach to reduce the overhead of copying data

between the user space and the kernel space, and hence to reduce the data processing delay.

Another major contribution of this paper is to identify and address the effects of data pipeline and MAC layer interference on the performance of caching. Although some researchers have addressed the effects of MAC layer interference on the performance of TCP and network capacity, this is the first work to study this problem in the context of cache management. We study the effects of different MAC layers, such as 802.11-based ad hoc networks and multi-interface-multichannel-based mesh networks, on the performance of caching. We also propose solutions for our asymmetric approach to identify the best nodes to cache the data. The proposed algorithm well considers the caching overhead and adapts the cache node selection strategy to maximize the caching benefit on different MAC layers. Our results show that the asymmetric approach outperforms the symmetric approach in traditional 802.11- based ad hoc networks by removing most of the processing overhead. In mesh networks, the asymmetric approach can significantly reduce the data access delay compared to the symmetric approach due to data pipelines.

•We propose the Asymmetric Cooperative Cache Approach on identifying the effects of data pipeline.

•We provide a greedy approach to determine the cache placement

Background Work

There have been several implementations of wireless ad hoc routing protocols. Royer and Perkins suggested modifications to the existing kernel code to implement AODV. By extending ARP, Desilva and Das presented another kernel implementation of AODV. Dynamic Source Routing (DSR) has been implemented by the Monarch project in FreeBSD. This implementation was entirely in kernel and made extensive modifications in the kernel IP stack. In , Barr et al. addressed issues on system-level support for ad hoc routing protocols. In , the authors explored several system issues regarding the design and implementation of routing protocols for ad hoc networks. They found that the current operating system was insufficient for supporting on-demand or reactive routing protocols, and presented a generic API to augment the current routing architecture. However, none of them has looked into cooperative caching in wireless P2P networks.

Caching Schemes in Wired Networks

Cooperative caching has been widely studied in the Web environment. These protocols can be classified as message based, directory-based, hash-based, or router based. Wessels and Claffy introduced the Internet cache protocol (ICP) , which has been standardized and is widely used. As a message-based protocol, ICP supports communication between caching proxies using a query-response dialog. It has been reported that ICP scales poorly with an increasing number of caches. Directory-based protocols for cooperative caching enables caching proxies to exchange information about cached content. The information is compressed using arrays of bits. Notable protocols of this class include Cache Digests and summary cache . The most notable hash-based cooperative caching protocol constitutes the cache array routing protocol (CARP) . The rationale behind CARP constitutes load distribution by hash routing among Web proxy cache arrays. Wu and Yu introduced several improvements to hash-based routing, considering network latency and allowing local replication. Web cache coordination protocol (WCCP), as a router-based protocol, transparently distributes requests among a cache array. These protocols usually assume fixed network topology and often require high computation and communication overhead. However, in an ad hoc network, the network topology changes frequently. Also, mobile nodes have resource (battery, CPU, and wireless channel) constraints and cannot afford high computation or communication overhead. Therefore, existing techniques designed for wired networks may not be applied directly to ad hoc networks.

Caching Schemes in Wireless Networks

Most of the previous research, in ad hoc networks focuses on routing, and not much work has been done on data access. Addressed the cooperation among sensor nodes during data collection. Applied the query-forwarding concept to sensor networks. They proposed a two-tier data dissemination (TTDD) model for wireless sensor networks. TTDD requires the construction of a grid structure in fixed sensor networks. The nodes at the grid crossing points work as routers, which forward queries to the source and forward data to the sink. Although both approaches use cache, their focus is on data aggregation and compression for sensor networks, not on cooperative caching and data access.

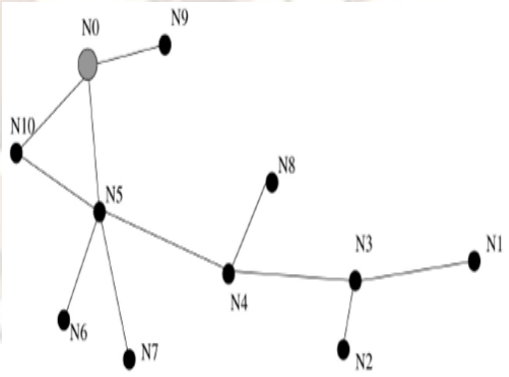

Fig.3: Wireless P2P Network

To effectively disseminate data in ad hoc networks, data replication and caching can be used. Data replication schemes in ad hoc networks have been studied by Har. However, these schemes may not be very effective due to the following reasons: First, because of frequent node movement, powering off or failure, it is hard to find stable nodes to host the

replicated data. Second, the cost of initial distribution of the replicated data and the cost of redistributing the data to deal with node movement or failure is very high. Similar to the idea of cooperative caching, Papadopouli and Schulzrinne proposed a 7DS architecture, in which a couple of protocols are defined to share and disseminate data among users that experience intermittent connectivity to the Internet. It operates either on a prefetch mode to gather data for serving the users' future needs or on an on demand mode to search for data on a single-hop multicast basis. Unlike 7DS architecture, they focus on data dissemination instead of cache management. Further, their focus is the single-hop environment instead of the multihop environment as in our work. To increase data accessibility, mobile nodes should cache different data items than their neighbors. Although this increases data accessibility, it also can increase query delays because the nodes might have to access some data from their neighbors instead of accessing it locally. In addition, replicating data from the server could create security problems. As Figure 2 illustrates, the cooperative cache- based framework stays on top of the routing protocols. It relies on several components, such as secure cooperative caching, cache management, and information search to provide services to the upper layer.

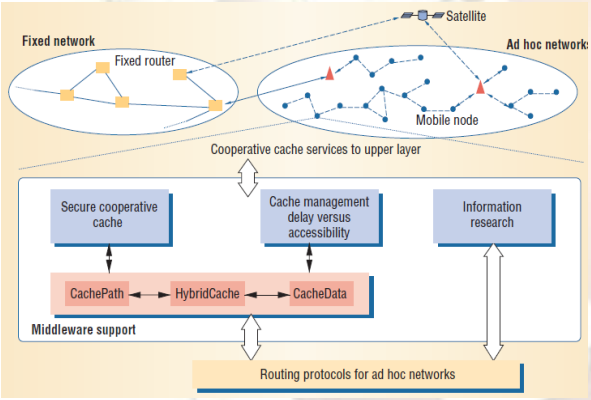

Fig.4: Cooperative cache-based data access framework

A cooperative caching scheme designed specifically for accessing multimedia objects in ad hoc networks. When a query comes, this scheme relies on flooding to find the nearest node that has the requested object. Using flooding can reduce the query delay since the request may be served by a nearby node instead of the data center faraway. Thus, it is good for multimedia applications which have strict delay requirements. Another benefit of using flooding is that multiple nodes that contain the requested data can be found. If the data size is very large, when the link to one node fails, the requester can switch to other nodes to get the rest of the requested data. Using flooding incurs significant message overhead. To reduce the overhead, flooding is limited to nodes within k hops from the requester, where k is the number of hops from the requester to the data center, but the overhead is still high. In a wireless network where nodes are uniformly distributed, on average, there are $\pi k2$ nodes within k-hops range of a mobile node. Therefore, $\pi k2$ messages are needed to find a data item using this method. Moreover, when a message is broadcast in the network, many neighbors will receive it. Even if the mobile node is able to identify and drop duplicated messages, each node still needs to broadcast the messages at least once to ensure full coverage. If a node has c neighbors, on average, the total number of messages needs to be processed is $c\pi k2$ although the message complexity is still O(k2), the constant factor may be very high, especially when the network density is high.

Existing & Proposed Schema

Existing schemes can be categorized to three
A. Cache Path
B. Cache Data
C. Hybrid Cache

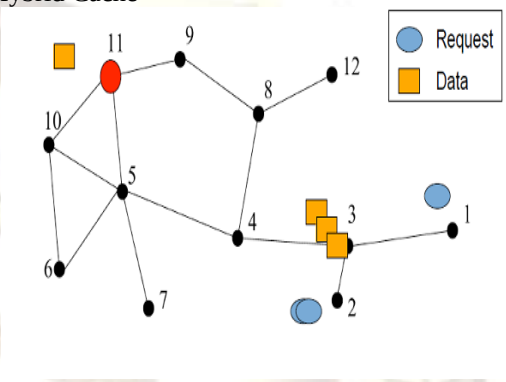

Fig.5: Wireless Network

Cache Path concept

Above fig.5 illustrates the CachePath concept. Suppose node N1requests a data item di from N11. When N3 forwards di to N1;N3 knows that N1 has a copy of the data. Later, if N2 requests di;N3 knows that the data source N11 is three hops away whereas N1 is only one hop away. Thus, N3 forwards the request to N1 instead of N4. Many routing algorithms (such as AODV and DSR) 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. However, mapping data items and caching nodes increase routing overhead, and the following techniques are used to improve CachePath's performance.

Cache Data concept

In CacheData, the intermediate node caches the data instead of the path when it finds that the data item is frequently accessed. For example, in Fig. 1, if both N6 and N7 request di through N5;N5 may think that di is popular and cache it locally. N5 can then serve N4's future requests directly. Because the Cache Data approach needs extra space to save the data, it should be used prudently. Suppose N3 forwards several requests for di to N11. The nodes along the path N3;N4, and N5 may want to cache di as a frequently accessed item. However, they will waste a large amount of cache space

if they all cache di. To avoid this, CacheData enforces another rule: A node does not cache the data if all requests for the data are from the same node. In this example, all the requests N5 received are from N4, and these requests in turn come from N3. With the new rule, N4 and N5 won't cache di. If N3 receives requests from different nodes, for example, N1 and N2, it caches the data. Certainly, if N5 later receives requests for di from N6 and N7, it can also cache the data.

Hybrid cache approach

CachePath and CacheData can significantly improve system performance. Our analysis showed that CachePath performs better when the cache is small or the data update rate is low, while CacheData performs better in other situations. To further improve performance, we propose HybridCache, a hybrid scheme that exploits the strengths of CacheData and CachePath while avoiding their weaknesses. Specifically, when a mobile node forwards a data item, it caches the data or path based on some criteria. These criteria include the data item size si and the time TTLi For a data item di , we use the some heuristics to decide whether to cache data or the path. HybridCache performs better than either approach because it applies either Cache Data or CachePath to different data items. HybridCache dynamically switches between CacheData and CachePath at the data item level, not at the database level.

Proposed methodology

To address the problem of the layered design, we propose an asymmetric approach. We first give the basic idea and then present the details of the scheme. Here the data replies are only transmitted to the cache layer at the intermediate nodes that need to cache the data. In this the data only reach the routing layer for most intermediate nodes, and go up to the cache layer only when the intermediate node needs to cache the data. Although the request message always needs to go up to the cache layer, it has a small size, and the added overhead is limited. If the requested data item is large, this asymmetric approach allows data pipeline between two caching nodes, and hence reduces the end-to-end delay. The cache layer processing overhead, especially data copying between kernel and user spaces, is also minimized because the data item is not delivered to the cache layer at the nodes that are unlikely to cache the data.

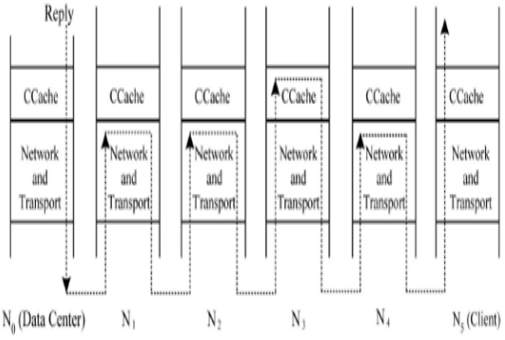

Fig.6: Asymmetric cache approach

Design & Analysis Of The Schema

A class icon is simply a rectangle divided into three compartments. The top most compartments contain the name of the class as shown in Fig.7. The middle compartment contains a list of attributes (member variables), and the bottom compartment contains a list of operations (member functions). In many diagrams, the bottom two compartments are omitted. Even when they are present, they typically do not show every attribute and operations. The goal is to show only those attributes and operations that are useful for the particular diagram.

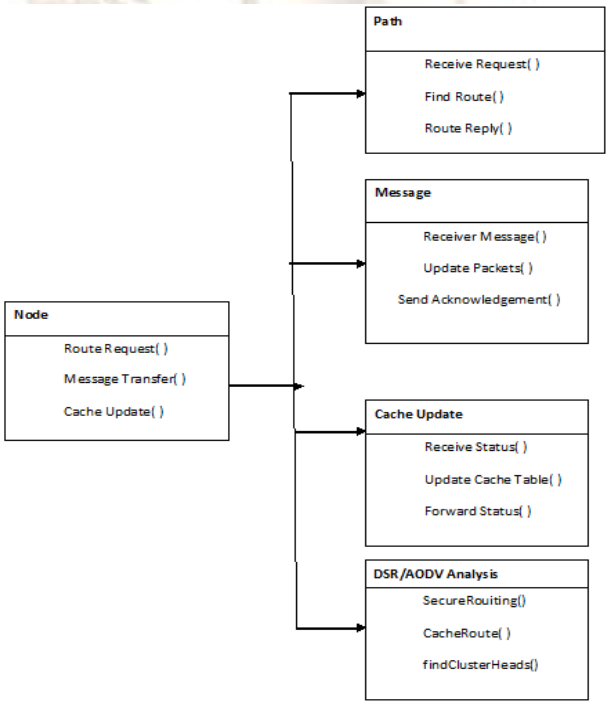

Fig.7: Interpretational Representation of the class Diagram of the Schema

CCSL is implemented as a user-space library. It implements common functions that cooperative cache schemes need and provides APIs to the Cooperative Cache Daemon (CCD).

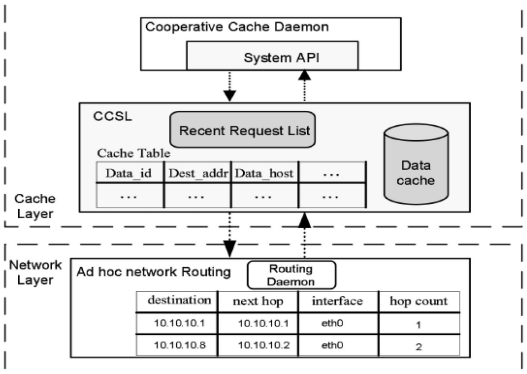

Fig.8: CCLS Design

The software architecture of CCSL. As shown in the fig.8, the Cache table is used to record data access. It is a hash table keyed by data id. Data items are cached in the data cache. Besides these two components, a list of recently received requests is maintained to detect duplicate data requests. If the data request is not a duplicate, it will be passed to the Cooperative Cache Daemon (CCD). An interface is provided between CCSL and the routing daemon. It enables CCSL to get the routing information which is used for transmitting cooperative cache layer packets.

Program of Node i
BEGIN
When a data item D_j passes by:
if local memory has available space and $(B_{ij} > \Theta)$
then cache D_j
else if there is a set D of cached data items such that (local benefit of $D < B_{ij} - \Theta$) and $(|D| \geq |D_j|)$, then replace D with D_j.
When a data item D_j is added to local cache
Notify the server of D_j.
Broadcast an `AddCache` message containing (i, D_j)
When a data item D_j is deleted from local cache
Get the cache list C_j from the server of D_j
Broadcast a `DeleteCache` message with (i, D_j, C_j)
On receiving an `AddCache` message (i', D_j)
if i' is nearer than current nearest-cache for D_j,
then update nearest-cache table entry and broadcast the `AddCache` message to neighbors
else send the message to the nearest-cache of i
On receiving a `DeleteCache` message (i', D_j, C_j)
if i' is the current nearest-cache for D_j
then update the nearest-case of D_j using C_j, and broadcast the `DeleteCache` message.
else send the message to the nearest-cache of i

Fig.9: A greedy cache placement algorithm

A greedy cache placement algorithm: To get the optimal cache placement, the data server needs to compute the aggregate delay for every possible cache placement set. Since there are 2^n (n is the length of the forwarding path) possible Ways to choose cache placement set, it takes $\Theta(2^n)$ which is too expensive. Therefore, we propose a greedy heuristic to efficiently compute the optimal cache placement.

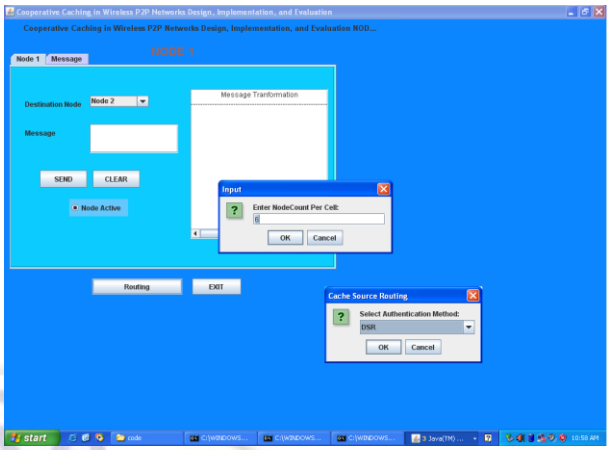

Fig.10: Specifying the Node count and Authentication Method

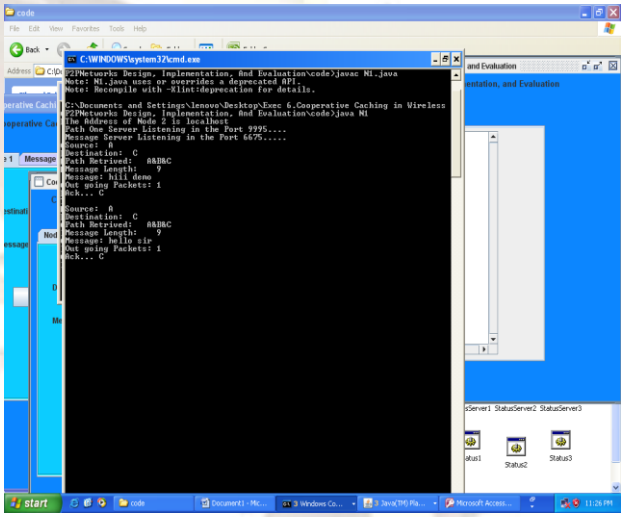

Fig.11: Addressing the nodes , Message and number of Outgoing Packets

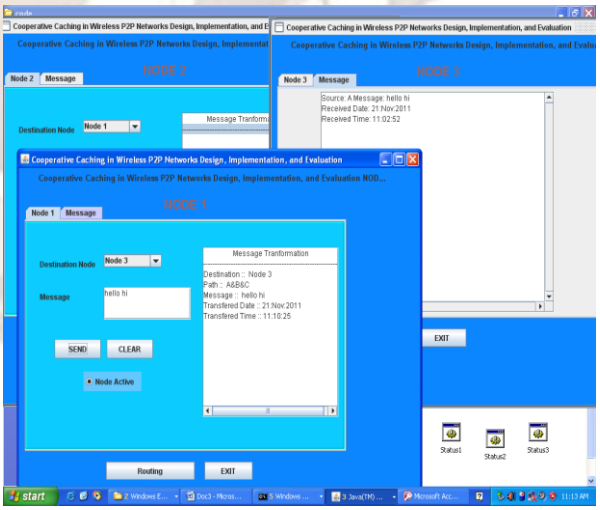

Fig.12: Message Transformation in P2P networking

Results

The below are the obtained results of the schema

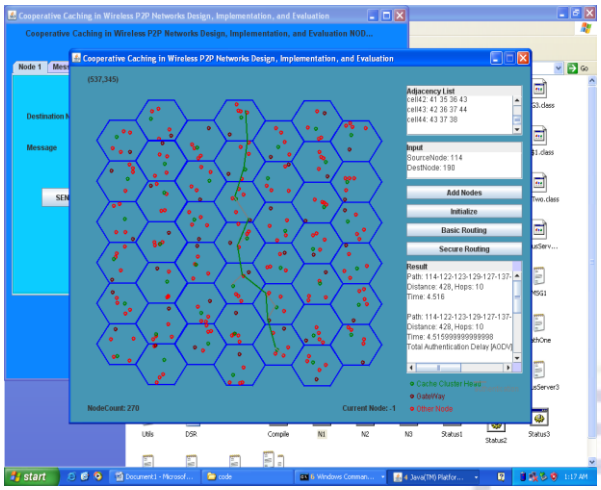

Fig.13: Final result of the Cooperative caching in wireless P2P networking

Conclusion & Future Enhancement

In this paper, we presented our design and implementation of cooperative cache in wireless P2P networks, and proposed solutions to find the best place to cache the data. In our asymmetric approach, data request packets are transmitted to the cache layer on every node; however, the data reply packets are only transmitted to the cache layer on the intermediate nodes which need to cache the data. This solution not only reduces the overhead of copying data between the user space and the kernel space, but also allows data pipeline to reduce the end-to-end delay. We evaluate our design for a large-scale network through simulations developed a prototype to demonstrate the advantage of the asymmetric approach. Since our prototype is at a small scale we evaluate our design for a large-scale network .Our results show that the asymmetric approach outperforms the symmetric approach in traditional 802.11-based ad hoc networks by removing most of the processing overhead. In mesh networks, the asymmetric approach can significantly reduce the data access delay compared to the symmetric approach due to data pipelines. To the best of our knowledge, this is the first work on implementing cooperative cache in wireless P2P networks, and the first work on identifying and addressing the effects of data pipeline .We believe many of these findings will be valuable for making design choices.

REFERENCES

[1] S. Desilva and S. Das, "Experimental Evaluation of a Wireless Ad Hoc Network," Proc. Ninth Int'l Conf. Computer Comm. And Networks, 2000.

[2] H. Eriksson, "MBONE: The Multicast Backbone," Comm. ACM, vol. 37, no. 8, pp. 54-60, 1994.

[3] L. Fan, P. Cao, J. Almeida, and A. Broder, "Summary Cache: A Scalable Wide Area Web CAche Sharing Protocol," Proc. ACM SIGCOMM, pp. 254-265, 1998.

[4] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The Impact of Multihop Wireless Channel on TCP Throughput and Loss," Proc. IEEE INFOCOM, 2003.

[5] IEEE, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Spec, IEEE 802.11 Standard, 1999.

[6] P. Kyasanur and N.H. Vaidya, "Routing and Link-Layer Protocols for Multi-Channel Multi Interface Ad Hoc Wireless Networks," ACM SIGMOBILE Mobile Computing and Comm. Rev., vol. 10, no. 1, pp. 31-43, 2006.

[7] W. Lau, M. Kumar, and S. Venkatesh, "A Cooperative Cache Architecture in Supporting Caching Multimedia Objects in MANETs," Proc. Fifth Int'l Workshop Wireless Mobile Multimedia, 2002.

[8] V. Kawadia, Y. Zhang, and B. Gupta, "System Services for Ad-Hoc Routing: Architecture, Implementation and Experiences," Proc. Int'l Conf. Mobile Systems, Applications and Services (MobiSys), 2003.

[9] E. Royer and C. Perkins, "An Implemenatation Study of the AODV Routing Protocol," Proc. IEEE Wireless Comm. And Networking Conf., 2000.

[10] B. Tang, H. Gupta, and S. Das, "Benefit-Based Data Caching in Ad Hoc Networks," IEEEE Trans. Mobile Computing, vol. 7, no. 3, pp. 289-304, Mar. 2008.

[11] J. Padhye, R. Draves, and B. Zill, "Routing in Multi-radio, Multi-hop Wireless Mesh Networks," Proc. ACM MobiCom, 2004.

[12] C. Perkins, E. Belding-Royer, and I. Chakeres, "Ad Hoc on Demand Distance Vector (AODV) Routing," IETF Internet Draft, draft-perkins-manet-aodvbis-00.txt, Oct. 2003.