IJCOT -Special Issue– The Malla Reddy National Conference on Information System and Knowledge Engineering (MRNC-ISKE 2013)- July 2013

Selfish Replica Allocation in Mobile Ad-hoc Network

T.Naga Vyshnavi*1, D.Veerabhadra Babu*2

M.Tech, Dept of CSE, SKDEC, Gooty, D.t: Anantapuram, A.P, India

M.C.A, M.Tech, (Ph.D), Associate Professor, Dept of CSE, SKDEC, Gooty, D.t: Anantapuram, A.P, India

ABSTRACT:

Some recent studies have shown that cooperative cache can improve the system performance in wireless P2P networks such as ad hoc networks and mesh networks. However, all these studies are at a very high level, leaving many design and implementation issues unanswered. In this paper, we present our design and implementation of cooperative cache in wireless P2P networks, and propose solutions to find the best place to cache the data. We propose a novel asymmetric cooperative cache approach 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. 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. We also 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 cooperative cache. Incentive mechanism tells how to encourage nodes in a peer-to-peer system to contribute their resources. A peer avoids contributing resources to the p2p system because of the factors like: cost of bandwidth, security reason as it has to open several ports in order to allow others to take out its resources and slowing down of self downloading process. In this paper we present a comparative study on some incentive models after going through several research papers in the line. A few models are implemented to show the simulation results. Finally conclusion is made by identifying the best incentive mechanism for p2p system and improvements are suggested based on the findings.

KEYWORDS

P2P, Incentive mechanism, Free-rider, White-washing, Wireless networks, P2P networks, cooperative cache.

I. INTRODUCTION

The peer-to-peer system is increasingly popular amongst internet users due to its nature of resource sharing ability, in which every peer node is contributing its resources to the network as well as consumes resources contributed by other peers. At the same time attacks on p2p networks also increased which become the threats to the existence of p2p system. The most common attacks are: free-riding and white-washing. Free-riders are peers in P2P network which do not contribute any resources but consume resources freely from the network. White-washers are free riders which frequently leave the system and re-appear with a different identity to get-rid-of penalties imposed by the network. In addition to these, some more attacks are: Sybil-attack and malicious behavior of peers. In Sybil attack [8], when an attacker entered into a system it creates a large number of identities in order to gain influence over the p2p system. The malicious behavior of peers says that a few number of peers may grouped together to cheat the system by increasing each others' points or grades within their small group. The existing incentive mechanisms can be classified into three categories such as schemes based on inherent generosity, monetary-based and reciprocity [1, 5]. In schemes based on inherent generosity every user decides whether to contribute or free-ride based on how its generosity compares to the current contribution cost in the system. If the social

generosity is below a threshold level, then numbers of free-riders are more and the system collapses. But if it exceeds the threshold limit, the contribution level increases with diminishing returns. For example warm-glow model which was based on inherent generosity, but it was not successful because it fails to explain the observed behavior of peer. Another example is the modeling framework that has been devised by M. Feldman et al. that studies the phenomenon of free-riding in P2P systems while taking user generosity into account. 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. 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 neighboring 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. It will save a large amount of battery power, bandwidth, and time if later accesses to the same data are served by the nearby soldier who has the data instead of the far away data center. As another example, people in the same residential area may access the Internet through a wireless P2P network, e.g., the Roofnet. After one node downloads a MP3 audio or video file, other people can get the file from this node instead of the far away Web server. Through these examples, we can see that if nodes are able to collaborate with each other, bandwidth and power can be saved, and delay can be reduced. Actually, cooperative caching which allows the sharing and coordination of cached data among multiple nodes, has been applied to improve the system performance in wireless P2P networks. However, these techniques [are only evaluated by simulations and studied at a very high level, leaving many design and implementation issues unanswered. There have been several implementations of wireless ad hoc routing protocols. In , Royer and Perkins suggested modifications to the existing kernel code to implement AODV. By extending ARP, Desilva and Das [7] 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 [2], 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. Although cooperative cache has been implemented by many researchers [6], [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 the multiple hop routing problem and cannot address the on-demand nature of the ad hoc routing protocols. To realize the benefit of cooperative cache, intermediate nodes along the routing path need to check every passing-by packet to see if the cached data match the data request. This certainly cannot be satisfied by the existing ad hoc routing protocols. In this paper, we present our 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 [10] 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.

II.RELATED WORKS

Caching is a key technique for improving data retrieval rate in both wired and wireless networks. The two basic types of cache sharing are push approach and pull approach. In push based cache sharing, a node broadcasts the caching update to all its neighbor nodes, on receiving a new data item. This updated information resides in the neighboring nodes for future use. Push based scheme improves the data availability at the cost of communication overhead. The disadvantage of the scheme is that an advertisement may become useless if no demand for the cached items occur in the vicinity. One more problem with the push based approach is that the caching information may not be used if the node moves out from the zone or due to cache replacement. These drawbacks are overcome with the pull based scheme. In case of pull based approach, a node broadcasts a request packet to all its neighbors, when it wants to access a new data item. If a neighbor has the requested data item it sends the data back to the requester node. The main disadvantage here is that, if the requested data item is not cached by any node in the neighborhood then the request originator must wait for the time out interval to expire before it resends the request to the data centre. This leads to access latency. Another drawback here is, if

more than one node have cached the requested data item then multiple copies will return to the requester which in turn will result in extra communication overhead. Duane Wessels and Kim Claffy, introduced the standardized and widely used Internet cache protocol(ICP). As a message-based protocol, ICP supports communication between caching proxies using a simple query-response dialog. Cache Digests [10] are a response to the problems of latency and congestion. Cache Digests support peering between cache servers without a request-response exchange taking place. A summary of the contents of the server (the Digest) is fetched by other servers which peer with it.

III. PROPOSED SYSTEM EVOLUTION

3.1 DESIGN AND IMPLEMENTATION OF COOPERATIVE CACHING

In this section, we first present the basic ideas of the three cooperative caching schemes proposed in: Cache Path, Cache Data, and Hybrid Cache. Then, we discuss some design issues and present our design and implementation of cooperative cache in wireless P2P networks.

3.2 Cooperative Caching Schemes

Fig. 1 illustrates the Cache Path concept. Suppose node N1 requests a data item di from N0. 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 N0 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 [20] and DSR [12]) 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 Cache Path's performance. In Cache Path, a node need not record the path information of all passing data. Rather, it only records the data path when it is closer to the caching node than the data source.

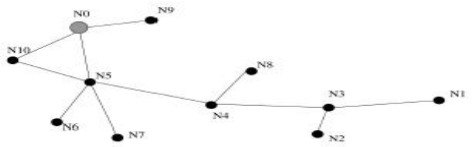

Fig. 1. A wireless P2P network.

For example, when N0 forwards di to the destination node N1 along the path N5 _ N4 _ N3;N4 and N5 won't cache di path information because they are closer to the data source than the caching node N1. In general, a node caches the data path only when the caching node is very close. The closeness can be defined as a function of the node's distance to the data source, its distance to the caching node, route stability, and the data update rate. Intuitively, if the network is relatively stable, the data update rate is low, and its distance to the caching node is much shorter than its distance to the data source, then the routing node should cache the data path. In Cache Data, 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 N0. 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, Cache Data 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.

3.3 Design Issues on Implementing

Cooperative Cache In this paper, we focus on design and implementation of the Cache Data scheme discussed in the above section. To realize the benefit of cooperative cache, intermediate nodes along the routing path need to check every passing-by packet to see if the cached data match the data request. This certainly cannot be satisfied by the existing ad hoc routing protocols. Next, we look at two design options.

3.3.1 Integrated Design

In this option, the cooperative cache functions are integrated into the network layer so that the intermediate node can check each passing-by packet to see if the requested data can be served. Although this design sounds straightforward, several major drawbacks make it impossible in real implementation. The network layer is usually implemented in kernel, and hence, the integrated design implies a kernel implementation of cooperative cache. However, it is well known that kernel implementation is difficult to customize and then it is difficult for handling different application requirements. Second, kernel implementation will significantly increase the memory demand due to the use of Cache Data. Finally, there is no de facto routing protocol for wireless P2P networks currently. Implementing cooperative cache at the network layer requires these cache and routing modules to be tightly coupled, and the routing module has to be modified to add caching functionality. However, to integrate cooperative cache with different routing protocols will involve tremendous amount of work.

3.3.2 Layered Design

The above discussions suggest that a feasible design should have a dedicated cooperative cache layer resided in the user space; i.e., cooperative cache is designed as a middleware lying right below the application layer and on top of the network layer (including the transport layer). There are two

options for the layered design. One naïve solution uses cross-layer information, where the application passes data request (search key) to the routing layer, which can be used to match the local cached data. However, this solution not only violates the layered design, but also adds significant complexity to the routing protocol which now needs to maintain a local cache table. Further, if an intermediate node needs to cache the data based on the cooperative cache protocol, it has to deal with fragmentation issues since some fragments of the data may not go through this node. Thus, this naive solution does not work in practice. Another solution is to strictly follow the layered approach, where the cooperative cache layer is on top of the network layer (TCP/IP). Fig. 2 shows the message flow (dashed line) in the layered design. In the figure, N5 sends a request to N0. Based on the routing protocol, N5 knows that the next hop is N4 and sends the request to N4 encapsulating the original request message. After N4 receives the request, it passes the message to the cache layer, which can check if the request can be served locally. This process continues until the request is served or reaches N0. After N0 receives the request, it forwards the data item back to N5 hop by hop, which is the reverse of the data request, as shown in Fig. 2b. Note that the data has to go up to the cache layer in case some intermediate nodes need to cache the data. Although this solution can solve the problems of the naïve solution, it has significant overhead. For example, to avoid caching corrupted data, reliable protocols such as TCP are needed. However, this will significantly increase the overhead, since the data packets have to move to the TCP layer at each hop. Note that the data packets only need to go to the routing layer if cooperative cache is not used. Further, this solution has a very high context switching overhead. At each intermediate node, the packets have to be copied from the kernel to the user space for cache operations, and then re-injected back to kernel to be routed to the next hop. The pipeline effect. Another problem of the layered design is the lack of data pipeline. Normally, the transport layer can fragment a large data item into many small data packets, which are sent one by one to the next hop. If there are multi-hop between the sender and the receiver, these small packets can be pipelined and the end-to-end delay can be reduced.

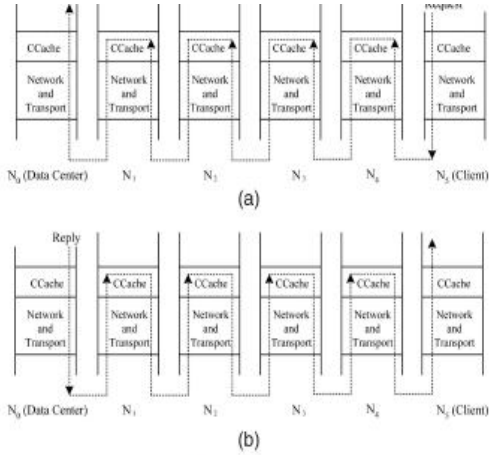

Fig. 2. Layered design. (a) The request packet flow and (b) the reply packet flow.

In cooperative cache, the caching granularity is at the data item level. Although a large data item is still fragmented by the transport layer, there is no pipeline due to the layered design. This is because the cache layer is on top of the transport layer, which will reassemble the fragmented packets. Since all packets have to go up to the cache layer hop by hop, the network runs like “stop and wait” instead of “sliding window.” This will significantly increase the endto-end delay, especially for data with large size.

IV. CONCLUSION

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 layeron 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 have 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 through simulations. Our simulation 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 and MAC layer interference on cache management. We believe many of these findings will be valuable for making design choices.

V. REFERENCES

[1] R. Agu¨ ero and J.P. Campo, "Adding Multiple Interface Support in NS-2," Jan. 2007.

[2] B. Barr, J. Bicket, D. Dantas, B. Du, T. Kim, B. Zhou, and E. Sirer, "On the Need for System-Level Suppport for Ad Hoc and Sensor Networks," ACM Operating System Rev., vol. 36, no. 2, pp. 1-5, Apr. 2002.

[3] J. Bicket, D. Aguayo, S. Biswas, and R. Morris, "Architecture and Evaluation of an Unplanned 802.11b Mesh Network," Proc. ACM MobiCom, 2005.

[4] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web Caching and Zipf-like Distributions: Evidence and Implications," Proc. IEEE INFOCOM, 1999.

[5] G. Cao, L. Yin, and C. Das, "Cooperative Cache-Based Data Access in Ad Hoc Networks," Computer, vol. 37, no. 2, pp. 32-39, Feb. 2004.

[6] M. Cieslak, D. Foster, G. Tiwana, and R. Wilson, "Web Cache Coordination Protocol v2.0," IETF Internet Draft, 2000.

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

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

[9] 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.

[10] 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.

T.Naga Vyshnavi M.Tech, SRI KRISHNA DEVARAYA ENGINEERING COLLEGE,
JNTU, Ananthapuram.

D.Veerabhadra Babu M.C.A, M.Tech (Ph.D),
Associate Professor, Department of CSE, SRI KRISHNA DEVARAYA ENGINEERING COLLEGE,
Gooty, Ananthapuram (Dt).