IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

STABILIZE THE MOVEMENT OF NODES ON ANYCAST ROUTING WITH JAMMING RESPONSIVE IN MOBILE AD-HOC NETWORKS

P.Vijayakumar*, A.Anusha Priya
* M.C.A., M.Phil Research Scholar in Computer Science, Muthayammal College of Arts and Science, Rasipuram,Tamilnadu,India.
M.C.A., M.Phil.,[Ph.D] Associate Professor in PG Computer Science, Muthayammal College of Arts and Science, Rasipuram,Tamilnadu,India.

ABSTRACT

Anycast is an important way of communication for Mobile Ad hoc Networks (MANETs) in terms of resources, robustness and efficiency for replicated service applications. Most of the anycast routing protocols for MANETs select unstable and congested intermediate nodes, thereby causing frequent path failures and packet losses. We propose Node movement Stability and Congestion aware Anycast Routing scheme in MANETs (NSCAR) that employs two models, namely (i) Node Movement Stability Model To Identify Stable Nodes, and (ii) Congestion Model That Considers Congestion Aware Parameters Like Channel Load And Buffer Occupancy. These models and Dynamic Source Routing (DSR) protocol are used in the route discovery process to select nearest k-servers that facilitate creation of stable paths from designated client to one of the nearest servers. A server among k-servers is selected based on route stability, channel load, hops, and server load. The simulation results indicate that proposed NSCAR demonstrates reduction in control overheads, improvement in Packet Delivery Ratio (PDR) and reduction in end-to-end delays compared to existing methods such as flooding and load balanced service discovery.

KEYWORDS: NSCAR, MANET, DSR, PDR.

INTRODUCTION

In wireless ad hoc networks, due to node mobility, the lifetime of routes can be very limited. Selecting stable routes can reduce rerouting times and the overhead of route discovery effectively, which is essential for Quality of Service (QoS) provisioning. Currently, the stability-based QoS routing algorithms, also called stable routing algorithms, have drawn extensive attentions. All stable routing algorithms use the path stability as routing criteria. The stability of a path can be determined according to a path stability computation method and the stability of each link along this path. The stability of a single link can be captured using various mobility models. Yet the path stability computation methods can considerably affect the performance of stable routing algorithms. If the assumption of link independence is adopted, then the stability of a path can be computed by multiplying the stabilities of all links along this path. The weakest link rule is applied, which defines the stability of a path to be the stability of the most instable link along this path. Recently, most stable routing algorithms adopt the Link Independence Assumption- Based Method (LIAM for short) or the Weakest Link Rule (WLR for short), which are the traditional path stability computation methods. However, the traditional methods have not considered the dependencies among links, and may be far from the real case. In fact, in ad hoc networks, the existence of wireless links is correlated.

In this paper, we consider the problem of congestion aware routing in Mobile Ad-Hoc Networks (MANETs). We propose a node stability-based location updating approach. In order to optimize the routing, most of the existing routing algorithms use some mechanism for determining a node's neighbors. This information is stored in a table called the Neighbor Table. The updating of the Neighbor Table is referred to as location updating. To evaluate our proposed algorithm, we simulated it and compared its performance with that of the performance of the conventional location updating algorithm, by considering different performance measures such as the number of collisions on the carrier, the number of acknowledgments received and the energy consumed. In our work, we obtained different types of results by varying different parameters such as the number of nodes and the terrain dimensions.

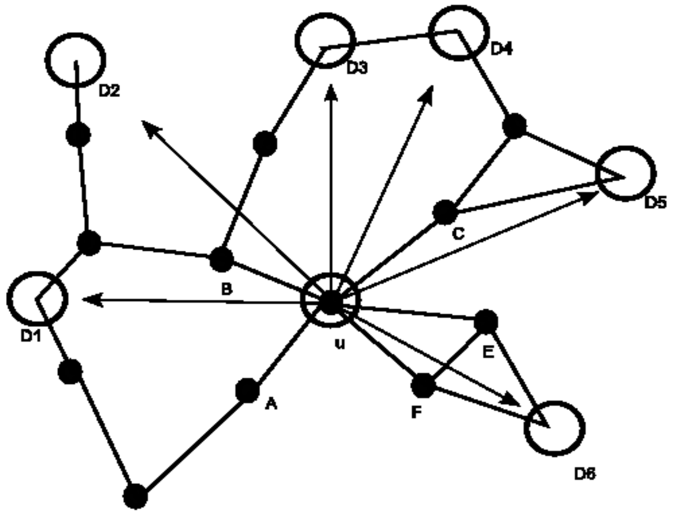

Figure 1. A sample Multi cast split node

While attempting to use the technique for multicasting, we identified several new issues that were not present in unicast routing algorithms.

RELATED WORK

A MANET is characterized as an infrastructure less cooperative engagement of mobile nodes, forming networks with continuously changing topologies. They do not have centralized network managers, access points, fixed base stations, or a backbone network for controlling the network management functions. They do not possess designated routers for making routing decisions. All the nodes in MANETs take part in routing by acting as routers for one another. However, several hops are normally needed in such networks for the transmission of data from one node to another, because of the limited wireless transmission range associated with the operation of the mobile nodes.

Anycasting is a network service that selects the best one of service providers in an anycast group as a destination. While anycasting offers better service flexibility in mobile ad hoc networks (MANETs), it also Increasing new problems. In MANETs, every node can move arbitrarily, and the routes from mobile nodes to their service providers would vary. Therefore, anycast service discovery in MANETs usually relies on network-layer message broadcasting, which leads to large traffic overhead for the scarce bandwidth of MANETs. In this work, we present a traffic-control scheme for anycast service discovery in MANETs. Our scheme can reduce the volume of query messages and the reply messages. In addition to basic anycasting, our scheme also supports k-anycast service that requests for k anycast service providers in each service instance. With k-anycast service, the fault tolerance and service flexibility of our scheme can be improved. Experimental results demonstrate that our scheme is efficient and feasible for MANETs.

In hierarchical tree-based routing for service discovery in a MANET, all routes form a tree infrastructure. Therefore, the transmitted packets from a node may go up to the tree root and down to the leaf node. The routing overhead of the tree-based routing algorithm cannot be avoided if the packet forwarding is based on parent-child relationships, even the destination node may be located near the source node. In order to improve such problem, each node should consider its neighbor nodes as next hop nodes in their routing tables. In this work, we present a shortcut any cast tree routing in MANETs. We minimize control overhead by establishing the routing tables only for certain nodes on an anycast tree. Our scheme can reduce the volume of query messages as well as the reply messages. It also reduces the transmission latency. We also improve the information accuracy of the routing tables by periodically transmitting update messages. As MANETs differ from the other infrastructure-based networks, in the sense that they do not have any fixed stations or routers to route the data packets, an efficient way the nodes can use to establish their network is to use a distributed routing approach. The previously mentioned characteristics of MANETs, particularly those arising due to their mobility, and the continuously changing network topologies, pose several challenges. Due to the continuously changing topologies, the routes that were once considered to be the "best" routes may no longer remain the same at a later time instant. This, therefore, requires a continuous re-computation of routes, and there is no permanent convergence to a fixed set of routes in such networks. So, any routing protocol that needs to operate in

MANET environments should take these issues into consideration. A number of unicast routing protocols have been proposed for use in MANETs for efficiently transmitting information. Many metrics have been proposed that can help evaluate the performance of the different MANET protocols. In order to perform routing in an efficient manner, many routing protocols (working in the network layer) require the mechanism below it (in the data link layer) to provide a table containing a list (and some additional information) of all nodes neighboring that node.

This is facilitated by a mechanism called Location Updating. As we have discussed, in MANETs, the nodes are mobile, which renders the network topology in MANETs susceptible to change with time. The frequency of the topology change is dependent on the application type. Since the locations of the nodes are susceptible to continuous changes, all nodes should have the information regarding the nodes to which each of them can directly communicate. To get the updated information of the neighboring nodes, a node requires location updating. The location updating process updates the information of the nodes' neighbors. In the conventional location updating algorithm, each node periodically broadcasts a Hello packet and on receiving the acknowledgment from the node, it updates its Neighbor Table. Our algorithm is based on the stability of the nodes. It updates the Neighbor Table of each node such that the more stable nodes, as preferred choices, send an acknowledgment. So, the network traffic will reduce, since a less number of acknowledgments are transmitted.

METHODS

LOCATION UPDATING

Since most of the prominent routing protocols used in ad-hoc networks are distributed in nature, it is quite likely that every node in the network will process routing information at some time or the other. All the uni-path routing protocols may be classified to be either table-based (proactive), or of an on-demand (reactive) nature. Table-based protocols are characterized by their ability to maintain routing tables that store information about routes from one node in the network to the rest of the others. Obviously, this requires that the nodes in the network maintain the table up-to-date by exchanging routing information between the participating nodes. Although, in general, the table-based protocols may be easy to implement, the major limitation with these protocols is that, due to the ethically highly mobile and dynamic nature of ad-hoc networks, the maintenance of routing information in these tables is challenging. On-demand routing protocols, on the other hand, alleviate the previous problems and make routing in them more scalable to highly dynamic and large networks. As the name suggests, on-demand routing protocols are characterized by the computation of routes on an "as-required" basis. In on-demand routing protocols, there is, initially, a route discovery phase, in which a route is found between two nodes. The route discovery phase is normally followed by a route maintenance phase in which a broken link in a route is repaired, or a new route found. In order to optimize the routing process, most of the routing algorithms use some mechanism for determining a node's neighbors. This information is stored in the Neighbor Table.

Our proposed algorithm improves upon the following factors:

- Less number of acknowledgments is transmitted in maintaining
- the Neighbor Table;
- Less number of collisions takes place;
- Less updating is required to maintain the Neighbor Table;
- Always, an updated Neighbor Table is available for each node.

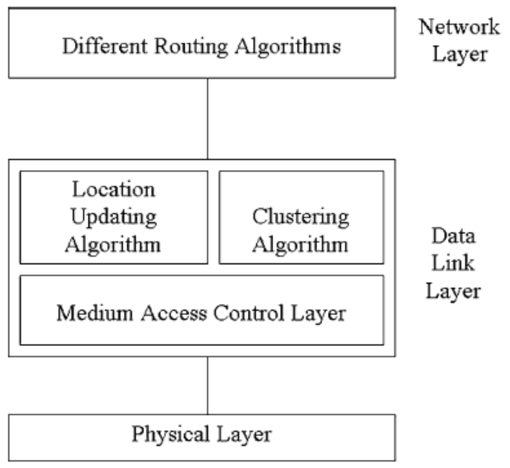
Different Routing Algorithms
Network Layer
Location Updating Algorithm
Clustering Algorithm
Medium Access Control Layer
Data Link Layer
Physical Layer

Figure 2. Position of Location updating in the protocol stack

In order to update the Neighbor Table, our algorithm uses Hello packets and Acknowledgments. Our main focus is to reduce the network traffic caused by the acknowledgement packets. We define the following terms in the manner given as follows.

1) Stability: The stability defined in the algorithm is the relative stability of one node with respect to another node. Let the stability of a node A with respect to the other node B be K1(K1>0)Then, it is not necessary that the stability of B with respect to A will also be K1. Stability of one node A with respect to another node B depends on the following factors.

A. How close the node A is to the boundary of the transmission range of node B. The closer the nodes A and B are, the more stable will be the node A.
B. The relative velocity of the two nodes. If they come closer, the stability increases. On the other hand, if they move apart from each other, it decreases.
C. Battery backup of A. If the time remaining for which the node can stay powered up is very less, the stability of A will decrease. This is because A can die out any time.
D. In case, the node A is stationary in its position and has good battery backup, then the node A will be considered to be stable. Our algorithm assumes the value of K to lie between K1 and K2, i.e., K1<=K<=K2 and (K1,K,K2)>0.

2) Retransmission Time: The retransmission time is the time after which the next Hello packet will be broadcasted. The retransmission time for a node in our algorithm is not fixed, but is dependent on the minimum value of the stability factor of its neighbors. If the neighbors are more stable, the retransmission time is increased, so that unnecessary Hello packets are not sent on the network. The retransmission time of a node depends on the stability of its neighbors, i.e., on the stability constant.
Our algorithm assumes the value of T to lie between T1 and T2. i.e., T1<=T<=T2 and (T1,T,T2)>0.

Consider the case when many receiving nodes are on the circumference of the circle defining the range of transmission of the sending node. This means that the retransmission time will be low. This further means that more frequent retransmissions will occur, leading to more traffic and more collisions in the network. But, since we have considered a lower limit on the retransmission time i.e., T1<=T<=T2, which in itself is higher than that used by the conventional algorithm, the number of packets sent and acknowledgement, received is always less than the conventional algorithm. This can be seen from the graph plotted. Consider the case where most of the nodes in the routing table have high stability condition and only one has lower stability.

In such a case, as only one node is unstable, there is a higher probability that this node will eventually move out of the range of the other nodes resulting in the increase of the Krecv value, which, in turn, will increase the retransmission time. If the node does not move out of the range of the other nodes, then it implies that this node has become, or is in the process of becoming stable and this will lead to an increase in Krecv value. If we have a different value of the retransmission time for each node, it means that a different Hello packet needs to be transmitted by each node and this will lead to an overall increase in the network traffic.

IMPLEMENTATION

Data Structures

This section describes the various data structures that are used for the implementation of the proposed node stability-based location updating algorithm. Hello Packet. This packet is broadcasted periodically after a certain retransmission time for updating the Neighbor table. This packet has various fields which are used by the nodes to calculate the stability factor. These fields are as follows.

- **Packet Type**: This field is used to specify the type of packet.
- **Source Address:** This field specifies the address of the node that transmits the Hello packet.
- **Destination Address:** This field specifies the address of the node that receives the Hello packet.
- **HPInfo**: This field contains various information about the node that transmits the Hello packet.
- Coordinates: This specifies the coordinates of the node.
- **Velocity Vector:** This specifies the velocity of the node.
- **Range:** This specifies the transmission range of the node.
- **Limiter K:** This specifies the value of K, below which each of the neighbor nodes has to send the acknowledgement.

Acknowledgement Packet

This packet is transmitted on receiving the Hello packet. This packet contains the following fields.

- **Packet Type**: This field specifies the type of packet.
- **Source Address**: This field specifies the address of the node that transmits the Hello packet.
- **Destination Address:** This field specifies the address of the node that receives the Hello packet.
- **HPACKInfo:** This field contains various information about the node that transmits the acknowledgement.
- **KSEND:** This field specifies the stability of the node with respect to the node that transmits the hello packet.

Neighbor Table

Neighbor table maintains the information about the various nodes which are in the vicinity of the current node. The node maintains a linked list to store information about its neighbors. The linked list is a singly linked list, in which the nodes of the list contain pointers to the next node.

The following fields exist in the Neighbor Table.

- **Node Address:** This specifies the address of the neighbor node.
- **Ksend**: This specifies the stability value which gets transmitted to the node upon receiving the Hello packet.
- **Krecv:** This specifies the stability value which is received from the node whose address is specified above.
- **Flag**: This field is used by the algorithm and it specifies whether to expect an acknowledgement from the current node or not.
- **Next**: This is a pointer which points to the next node in the neighbor table.

Modules Applied

Following are the functions used in our simulation of the algorithm. These functions are used in maintaining the neighbor table.

- **MacLUTransmitHELLO**: This is a function which transmits the Hello packet. The Hello packet is broadcasted on the network.
- **Process HP**: This function is used whenever a Hello packet is received by a node. After receiving the Hello packet, it is processed to update the Neighbor Table and send the acknowledgment.
- **MacLUStartTimer**: This is used to start the timer. Whenever the timer expires, this function is called to restart the timer.
- **MacLUInit**: This is used to initialize each node. It is only called at the start of the simulation. Different variables in the nodes are initialized.
- **MacLUFinalize**: This is used to finalize each node. It is used at the end of the simulation. This function is called for each node. It is used to print the statistical information of each node.
- **MacLUTransmitHP ACK**: This is used to transmit the acknowledgment after receiving the Hello packet.
- P**rocess HP Ack**: This is used to process the received acknowledgment.

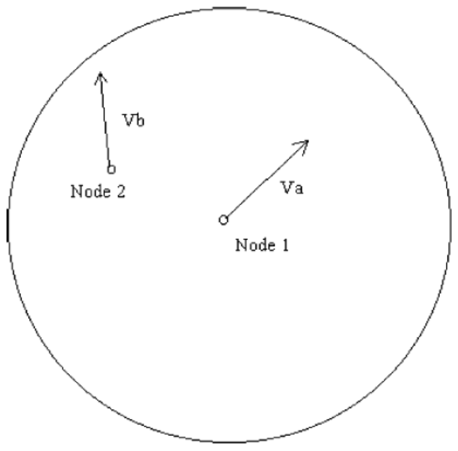

Fig. 2. Position of nodes with the direction of movement.

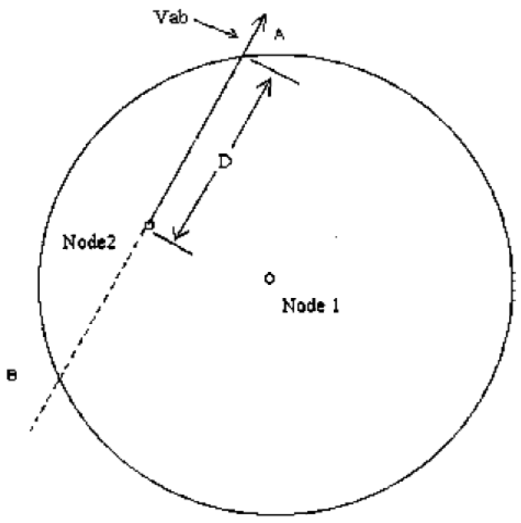

Fig. 3. Calculation of D.

Whenever an acknowledgment is received, the Neighbor Table is updated to indicate the presence of the node, from which the acknowledgment is received.

- **lu update nt:** When the Hello packet is transmitted, the nodes in the Neighbor Table are flagged to distinguish the node for which the acknowledgment is expected. When the timer is out and the acknowledgment is not received, this function is used to update the table, i.e., to remove the nodes from which acknowledgment was expected, but was not received.
- **lu calc k**: This is used for the calculation of the stability constant according to the different relative parameters such as the mobility of the node, the range of the node and their battery power.
- **Mark Nbr Table**: When the Hello packet is transmitted, this function is used to mark the nodes in the table from which the acknowledgment is expected. The nodes are marked on the basis of (Limiter).We refer to its stability with respect to that of Node1.

The arrows show the direction of movement of the node.
A and B: The intersection points of the relative velocity vector with the circle representing the range of the node which is transmitting the Hello packet.
D: The distance required to be covered by Node2 to get out of the range of Node1.
The simulation results demonstrate an encouraging support for the proposed algorithm.

The performance of our algorithm was tested on the following parameters:

- Number of Hello packet acknowledgements transmitted.
- Energy consumed.
- Number of collisions.

We infer from the set of results we obtained that the proposed algorithm works efficiently. It is capable of efficiently decreasing the amount of energy consumption, the number of Hello packet acknowledgements transmitted and the number of collisions in the network.

EXPERIMENTAL RESULTS

A closer investigation of our algorithm would reveal that the main focus in our approach is to reduce the number of acknowledgments by varying the updating information of less stable nodes more frequently than the more stable nodes.

Following are the advantages that should result from the likely less number of acknowledgments that get generated:
• Efficient use of bandwidth.
• Power conservation.

Because of the variable retransmission time, the following advantages should result:
• There should be power conservation because of the reduction in the probability of collision.
• There should be power conservation because of a reduced number of Hello messages.

Calculation of Stability Constant
In this section, we discuss how the stability constant is computed. In Figs. 2 and 3, let us refer to the following notations.
Node1: Let us assume that this is the node which broadcasts a Hello packet. The circle represents the range of the Node1.
Node2: Let us assume that Node2 represents the node which receives the Hello packet and is expected to send the Acknowledgment.

CONCLUSION AND FUTURE ENHANCEMENTS

Our algorithm works on the stability of the nodes. The stability of a node is determined by the combination of the speed of the node, the range of the node, the location of the node and the battery power of the node. The stability is not constant between two nodes, but changes with the condition such as velocity or battery power. The main focus of the proposed algorithm is to reduce the number of acknowledgment packets by varying the updating information of less stable nodes more frequently compared to the more stable nodes.

The proposed algorithm assumes some additional computations to be performed and some battery power is consumed while doing these computations. This is one of the limitations of our algorithm. We are, currently, investigating this issue and we aim to propose a better algorithm with lower overhead in the future. In the future, we are also aiming to investigate the problem by considering other performance parameters. Another limitation of our work is the issue of scalability in terms of the number of nodes in the network that we considered. When we increase the number of nodes in the network the execution time of our algorithm increases exponentially. In the future, we plan to investigate this issue as well and propose a better algorithm with respect to the execution time.

REFERENCES

[1] C. Plaxton, R. Rajaraman, and A. Richa, “Accessing nearby copies of replicated objects in a distributed environment,” in Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures, 1997.
[2] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, “Web caching and zipf-like distribution: Evidence and implications,” in Proceedings of the IEEE INFOCOM, 1999.
[3] D. R. Karger, E. Lehman, F. T. Leighton, R. Panigrahy, M. S. Levine, and D. Lewin, “Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web,” in Proceedings of the ACM Symposium on Theory of Computing, May 1997, pp. 654–663.
[4] A. Rowstron and P. Druschel, “Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems,” in Proceeding of the Middleware, 2001.
[5] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A scalable content addressable network,” in Processings of the ACM SIGCOMM, 2001, pp. 161172.
[6] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, “Chord: A scalable peer-to-peer lookup protocol for internet applications,” IEEE/ACM Transactions on Networking, vol. 11, no. 1, pp. 17-32, 2003.
[7] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz, “Tapestry: A resilient global-scale overlay for service deployment,” IEEE Journal on Selected Areas in Communications, vol. 22, no. 1, pp. 41–53, January 2004.
[8] Napster. [Online]. Available: http://www.napster.com.
[9] F. Dabek, B. Zhao, P. Druschel, and I. Stoica, “Towards a common api for structured peer-to-peer overlays,” in Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS 2003), Berkeley, California, USA, February 20-21 2003.

[10] B. Karp, S. Ratnasamy, S. Rhea, and S. Shenker, “Spurring adoption of dhts with openhash, a public dht service,” in Proceedings of the 3rd International Workshop on Peer-to-Peer Systems (IPTPS 2004), Berkeley, California, USA, February 26-27 2004.
[11] P. Maymounkov and D. Mazi`eres, “Kademlia: A peer-to-peer information system based on the xor metric,” in Processings of the IPTPS, Cambridge, MA, USA, February 2002, pp. 53–65.
[12] D. Malkhi, M. Naor, and D. Ratajczak, “Viceroy: a scalable and dynamic emulation of the butterfly,” in Processings of the ACM PODC’02, Monterey, CA, USA, July 2002, pp. 183–192.
[13] P. Francis, “Yoid: Extending the internet multicast architecture,” unpublished April 2000. http://www.aciri.org/yoid/docs/index.html.
[14] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, “Oceanstore: An architecture for global-scale persistent storage,” in Processings of the ACM ASPLOS, November 2002. A SURVEY AND COMPARISON OF PEER-TO-PEER OVERLAY NETWORK SCHEMES 21.
[15] W. J. Bolosky, J. R. Douceur, D. Ely, and M. Theimer, “Feasibility of a serverless distributed file system deployed on an existing set of desktop pcs,” in Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, June 2000, pp. 34–43.
[16] M. Waldman, A. D. Rubin, and L. F. Cranor, “Publius: a robust, tamperevident, censorship-resistant, web publishing system,” in Processings of the Ninth USENIX Security Symposium, Denver, CO, USA, 2000.
[17] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, “Widearea cooperative storage with cfs,” in Proceedings of the eighteenth ACM symposium on Operating systems principles, 2001, pp. 202–215
[18] R. Cox, A. Muthitacharoen, and R. Morris, “Serving dns using chord,” in Proceedings of the First International Workshop on Peer-to-Peer Systems, March 2002.