Optimal Caching with Content Broadcast in Cache-and-Forward Networks

Lijun Dong Dan Zhang Yanyong Zhang Dipankar Raychaudhuri
WINLAB, Rutgers University
671 Route 1 South
North Brunswick, NJ 08902-3390
{lijdong, bacholic, yyzhang, ray}@winlab.rutgers.edu

***Abstract*—With the rapid advance in the technology area of data storage, storage capacities have increased substantially while the price has been dropping fast. Motivated by this trend, it has been proposed in the Cache-and-Forward architecture that storage is incorporated into each intermediate CNF router. Contents can be cached at CNF routers when they flow through the network, and therefore, routers can serve the subsequent requests later on, without forwarding the requests to the host server, we refer to this caching paradigm as in-network caching. In this paper, the content caching is enhanced by Content Broadcast(CB), by which a CNF router broadcasts the information of cached contents to its neighboring nodes. In order to solve the problem that with limited storage, how an intermediate CNF router optimally decides which passing content should be cached, we develop a mathematical model for CB to minimize the average content retrieval latency, and propose the Independent Allocation algorithm. We compare the average content retrieval latencies of the proposed caching scheme with two other common cache replacement policies. We study the impact of cache size and the locality parameters. The proposed scheme is shown to provide significant performance improvement under various settings by as large as 65%.**

I. Introduction

The overwhelming use of today's network is for an end user to acquire a named chunk of data. Efficient content dissemination is becoming extremely important. Last few years, we have witnessed dramatic advances in the technology areas of microprocessor and data storage. For example where wireless access rates have increased 50-fold in the last decade, solid-state storage capacities have increased 200-fold, while dropping in cost to $2/GB. Since applications become more demanding, and new technology makes available larger storage, higher bandwidth, as well as diverse means of connecting to the Internet, a new networking paradigm can be made in protocol design for content delivery.

Cache-and-Forward (CNF) [1] has been proposed as a clean-slate architecture for next generation Internet that leverages the rapidly decreasing memory costs to provide in-network storage at routers. A detailed protocol description of CNF can be found in [2]. Fundamental to CNF architecture are two components: a transport layer service that operates in a hop-by-hop store-and-forward manner with large contents, and a caching scheme that integrates caching into each individual router to reduce network traffic and speed up content dissemination. In this paper, we focus on this new caching paradigm, which we call *Integrated In-Network Caching*. A straightforward in-network caching approach is to have each en-route CNF router independently decide whether or not to cache passing contents which we called *Cache-n- Capture*. When a request is routed through the router later, the router can "capture" the request and reply with the cached copy of the content, instead of forwarding the request to the original hosting sites.

However, Cache-n-Capture does not provide adequate performance because a CNF router is not aware of what other routers have cached. This unawareness can result in several undesirable situations. To name a couple, same content can be cached at neighboring routers, leading to a low neighborhood cache utilization; a router may unnecessarily forward a content request to the home server while its neighbor could have a copy of the content, leading to a longer retrieval latency. To address this unawareness, we advocate that each router should advertise the cached content to its neighbors, which is similar to the idea of summary cache proposed in [3]. We call this scheme to be *Content-Broadcast*(CB), i.e., it broadcasts the cached content information within the caching node's vicinity to help route requests to nearby cached copies. Although each CNF router has the ability to cache contents routed through, the capacity can not be unlimited large. In order to solve the problem of low neighborhood cache utilization, and achieve the minimum average content retrieval latency, we formulate a novel mathematical model which takes into account the content caching ability and enhanced content broadcast strategy of each intermediate CNF router. We propose the Independent Allocation algorithm to provide an optimal caching scheme with CB enabled.

The rest of the paper is organized as follows. We first give a brief overview of the related work in Section II. Next, we discuss the CB strategy in a CNF network. The proposed mathematical model and Independent Allocation algorithm are presented in Section III. Additionally, we conduct a set of performance studies, and showed the superiority of the optimal caching scheme with CB in Section IV. Finally, we provide concluding remarks in Section V.

II. Related Work

The idea of having Internet routers cache passing data has been discussed in several contexts. For example, in [4], the authors proposed to associate caching with en-route routers to speed up object access. In [5], the authors studied where to place caches for such a system. In [6], a scheme was proposed to dynamically place the object in the caches on the path from the server to the client in a coordinated fashion. The similar idea was also discussed in the context of Active Networks [7]. and [8], and in Active Reliable Multicast [9].

However, in the previous works, caches have not been considered as an integral part of the underlying network in the same way routers have been. Thus there has been no need to extend the existing routing protocols with content related information. Although in Summary cache proposed in [3], each proxy keeps a summary of the URLs of cached contents represented by a bloom filter, the proposed CNF architecture builds a different in-network caching framework, where each CNF router broadcasts the cached content information to neighboring nodes instead of the participating proxies, which reduces the overhead induced by content broadcasting. To our knowledge, with CB, this paper is the first to lay the mathematical groundwork for the physical problem to be meaningfully formulated, and for algorithms to be rigorously derived.

III. Mathematical Model for Content Broadcast

With caching enabled, each CNF router is able to cache passing contents selectively. However, requests may miss the CNF router that has the requested content in the cache if it is not on the routing path from the requester to the original server, even though the router might be much closer to the requester. If the requester gets to know the closer content location, the retrieval latency can be significantly reduced. Thus we propose that a CNF router explicitly advertises the information of a cached content to its neighbors, which might be propagated to a larger region, (i.e., CB). Take the scenario shown in Fig.1 as an example. The requester is trying to get a content from the web server by sending out a query packet. The requester and the web server are connected by five intermediate CNF routers. We suppose the CNF router B already caches this requested content. Without CB, the CNF router A simply forwards the query packet towards the web server, which is 6-hop away from the requester. However, with CB enabled at router B, router A is able to learn that there is a copy of the requested content cached by a nearby neighbor. A can forward the query packet directly to B, which substantially reduces the retrieval latency as seen by the requester.

Although by using CB, the content retrieval latency is reduced due to the knowledge of the locations of the nearby cached copies to the intermediate routers, we are still going to face the problem that as to which contents should be cached by a CNF router with the limited storage. In the following section, we model the problem by considering the influence of CB to the caching decisions made by each CNF router. We formulate an optimization problem in order to minimize the average content retrieval latency.

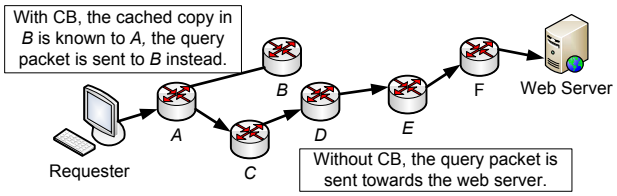

Fig. 1. A scenario with CB enabled

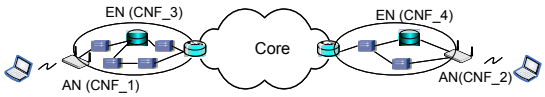

Fig. 2. CNF network

A. Optimization Problem Formulation

We envision that the CNF network adopts a tiered structures shown in Fig 2. In the core are high-bandwidth static routers. Outside the core are access networks, which are attached to a subset of core nodes. An access node(AN) is an aggregation point for mobile end nodes connected to it, which acts as the representative for content requesting. Meanwhile an end node(EN) is attached to wired terminals for the same role. Therefore we consider that content caching and request routing happen in the access networks and the core network. We model the access networks plus the core network as an undirected graph $G = (V, E)$, where a vertex in V represents a node (CNF router), and an edge in E represents a network link.

We assume that the popularity distribution of the contents is known a priori, which follows the Mzipf distribution [10]. Mzipf defines the probability of the i-th content being requested out of F available contents as

$$Pr(i) = \frac{1}{(i+q)^{\tau} \cdot K}, \tag{1}$$

and

$$K = \sum_{i=1}^{F} \frac{1}{(i+q)^{\tau}}, \tag{2}$$

where τ is the skewness factor which is the same as the skewness factor in Zipf distributions, and q is the plateau factor which controls the plateau shape (i.e. flattened head) near the most popular objects that are lowest ranked. A larger q value indicates a more flattened head.

We assume there are N nodes, denoted as $1, 2, \ldots N$ and F contents, labeled as $1, 2, \ldots F$. Each content file j has only one original server, which is one of the N nodes(CNF routers) and denoted as S_j. We define C_i as the set of contents originally hosted by node i , and C_i as the size of this set. For simplicity of exposition, we assume that all links in the network have the same bandwidth of B Mbps. But this assumption can be easily extended to take each link capacity into consideration. We fix the size of request packet size to be Q bits. The content size f_j, $j = 1, 2, \ldots, F$, can be different. The total round

delay of requesting content i over a hop is $D_j = D_q + D_p + d_j$, including the fixed process delay at an intermediate node, which is a constant D_p, and fixed per-hop request transmission delay $D_q = Q/B$, as well as the per-hop content transmission delay $d_i = f_j/B$. The content retrieval latency is proportional to the number of hops between the requesting node and the node that satisfies the request.

In addition, the following variables are defined:

- $P_{i,j}$, the probability that a content request is generated, which is from node i to request content j;
- $V_{i,j}$, indicator whether node i has $(= 1)$ content j in its cache or not $(= 0)$, or the caching probability;
- R_i, the storage limit of node i;
- $H_{a,b}$, the hop count of the shortest path from node a to node b;
- T_a, the maximum hop count between node a and any other node in the network;

With CB we make the assumption that any node knows exactly which content files are in the possession of any other node. When it needs a file, it goes straight to the nearest neighbor with the file. We are then interested in the average delay incurred herewith. These operations are well defined and they lead to a binary optimization problem. Let $B_{i,j}$ be the nearest neighbor to node i that has file j in the cache, the problem can be formulated as

$$\min \quad D = \sum_{i=1}^{N}\sum_{j=1}^{F} P_{i,j} H_{i,B_{i,j}} D_j, \tag{3}$$

$$\text{s.t.} \quad \sum_{j=1}^{F} V_{i,j} f_j \le R_i, \tag{4}$$

$$V_{S_j,j} = 1, j = 1, 2, \dots F, \tag{5}$$

$$V_{i,j} \in \{0, 1\}, \quad \forall i, j, \tag{6}$$

Our objective is to minimize the average content retrieval latency represented by D in Equation (3). The first constraint is to ensure that the total size of the contents cached on each router will not exceed its storage limit. The second constraint states that the probability that content j resides in its original server should be 1. Although $B_{i,j}$ itself is complicatedly connected to $V_{i,j}$, $H_{i,B_{i,j}}$ has, however, an analytic expression

$$H_{i,B_{i,j}} = \sum_{h=1}^{T_i} h U_{i,j}^h \prod_{k=0}^{h-1} (1 - U_{i,j}^k) \tag{7}$$

where $U_{i,j}^h$ is defined as the probability that node i can get the requested content j from one of its h-hop neighbors. It is given by

$$U_{i,j}^h = 1 - \prod_{\ell \in M_h^i} (1 - V_{\ell,j}). \tag{8}$$

Equation (7) comes from the observation that $B_{i,j}$ is the nearest neighbor of i holding content j if and only if any node i' strictly less than $H_{i,B_{i,j}}$ hops away must have $V_{i',j} = 0$. $U_{i,j}^h$ is an indicator whether at least one of node i's h-hop neighbors has content j in the cache. With Equation (7), Equation (3) becomes a standard binary optimization problem. Unfortunately, it is still hard to solve. To its remedy, we relax the constraint on $V_{i,j}$ such that it takes on a continuum of values in the interval $[0, 1]$:

$$\min \quad D = \sum_{i=1}^{N}\sum_{j=1}^{F} P_{i,j} \sum_{h=1}^{T_i} h D_j U_{i,j}^h \prod_{k=0}^{h-1} (1 - U_{i,j}^k), \tag{9}$$

$$\text{s.t.} \quad \sum_{j=1}^{F} V_{i,j} f_j \le R_i, \tag{10}$$

$$V_{S_j,j} = 1, j = 1, 2, \dots F, \tag{11}$$

$$0 \le V_{i,j} \le 1, \quad \forall i, j. \tag{12}$$

The relaxation has a physical interpretation, i.e., each node i caches files j with a probability of $V_{i,j}$. Meanwhile, it converts a hard binary program to a regular nonlinear program with linear constraints, for which we can derive a solution. Due to relaxation, the optimal value of (9) is even smaller than that of (3). In practice, each CNF server still has to make a binary decision as to cache or not, based on the solution to (9). Nonetheless, the solution in fact puts most of the optimal variables to either 0 or 1, as we will show later. Therefore, the relaxation does not strongly obscure the optimal binary solution.

A suboptimal solution to the nonconvex program in (9) can be obtained by considering the gradient, for which we compute

$$g_{i,j} = \frac{\partial D}{\partial V_{i,j}} \tag{13}$$

When i and j are fixed, $V_{i,j}$ exists in equation of D only when the request for content j is originated from any node p out of the total N nodes, meanwhile i is one of node p's h-hop away neighbors. h can be from 1 to T_p, which is the maximal number of hop count between node p and any other node in the network. In that case, $V_{i,j}$ will appear in D in two ways. One way is that node i is one of neighboring nodes which are $H_{p,i}$ hops away, and there is at least one copy of content j residing in this set of nodes, resulting in $U_{p,j}^{H_{p,i}}$ is positive. The other way is that content i is not cached in any of the nodes which are 1-hop till $H_{p,i}$ hops away and the content request is satisfied by a node that is further away.

Therefore,

$$g_{i,j} = \sum_{p=1}^{N} P_{p,j} H_{p,i} D_j \prod_{k=0}^{H_{p,i}-1} (1 - U_{p,j}^k) \prod_{\ell \in M_{H_{p,i}}^p \setminus \{i\}} (1 - V_{\ell,j}) \tag{14}$$

$$- \sum_{p=1}^{N} P_{p,j} \sum_{h=H_{p,i}+1}^{T_p} h D_j U_{p,j}^h \prod_{\substack{k=0 \\ k \ne H_{p,i}}}^{h-1} (1 - U_{p,j}^k) \prod_{\ell \in M_{H_{p,i}}^p \setminus \{i\}} (1 - V_{\ell,j}).$$

B. Independent Allocation Algorithm

In this section, we propose the Independent Allocation algorithm to give a solution to the optimization problem (9). As its name suggested, each node i in the network independently adjusts the contents cached locally. We assume

$V_{i',j}$, $i' \neq i$, $j = 1, 2, \ldots, F$, to be feasible (meeting the respective constraints) and known and fixed, hence the only variables are $V_{i,j}$, $j = 1, 2, \ldots, F$, which are local to node i. The objective function in (9) can be rewritten exclusively for node i

$$\min \quad \sum_{j=1}^{F} g_{i,j} V_{i,j} + \text{constant}, \tag{15}$$

Though (14) that gives the formula of $g_{i,j}$ is fairly complicated, it is clear that increasing $V_{i,j}$ for any particular choice of i and j, while keeping other caching probabilities fixed, can only decrease the average latency, i.e., caching a new file without replacing any old files can only reduce the latency. Apply this observation to (15), we find this is possible only when $g_{i,j} \leq 0$.

If $\sum_{j=1}^{F} f_j \leq R_i$, then the optimal solution to (15) is trivially $V_{i,j} = 1, \forall j$. Assume $\sum_{j=1}^{F} f_j > R_i$, and form the partial Lagrangian of (15)

$$\min \quad \mathcal{L} = \sum_{j=1}^{F} g_{i,j} V_{i,j} + \mu \left(\sum_{j=1}^{F} f_j V_{i,j} - R_i \right) \tag{16}$$

$$= \sum_{j=1}^{F} f_j (g_{i,j}/f_j + \mu) V_{i,j} - \mu R_i \tag{17}$$

subject to the other two constraints in (10) and (11), with $\mu \geq 0$ being the dual variable. Suppose the optimal dual variable μ satisfies $\mu = 0$. Since we know $g_{i,j} \leq 0$, the optimal solution is clearly $V_{i,j} = 1, \forall j$. But by assumption $\sum_{j=1}^{F} f_j V_{i,j} = \sum_{j=1}^{F} f_j > R_i$, i.e., the optimal solution violates the constraint. This contradiction shows that $\mu > 0$. Then the optimal solution is obtained by setting $V_{i,j} = 1$ if $S_j = i$ and setting

$$V_{i,j} = \begin{cases} 1, & g_{i,j}/f_j + \mu < 0, \\ 0, & g_{i,j}/f_j + \mu > 0, \\ \text{any feasible value}, & g_{i,j}/f_j = 0, \end{cases} \tag{18}$$

for those j such that $S_j \neq i$. Besides, by the complementary slackness of KKT, we have $\sum_{j=1}^{F} V_{i,j} f_j = R_i$. Together with 18, the optimal μ and $V_{i,j}$ can be solved.

The discussion above leads to the Independent Allocation algorithm, which consists of each node i randomly or periodically executing the following procedures:

1) Calculate the gradient $g_{i,j}$ for those contents that are not generated by the node i according to Equation (14). The number of such contents for node i is

$$w_i = F - O_i \tag{19}$$

Since the storage room for those contents must be saved on node i, there is no need to calculate their gradients.

2) Sort $g_{i,j}/f_j$ in the order of increasing order, for those j such that $i \neq S_j$:

$$g_{i,j_1}/f_{j_1} \leq g_{i,j_2}/f_{j_2} \leq \cdots \leq g_{i,j_{w_i}}/f_{j_{w_i}} \tag{20}$$

3) Reallocate $V_{i,j}$ to fill up the cache on node i. Find $0 \leq x \leq w_i$ such that

$$\sum_{c \in C_i} f_c + \sum_{1 \leq m \leq x} f_{j_m} \leq R_i \tag{21}$$

and

$$\sum_{c \in C_i} f_c + \sum_{1 \leq m \leq (x+1)} f_{j_m} > R_i \tag{22}$$

If $x = 0$, set $V_{i,j} = 0$, $\forall j$, i.e., there is no extra room for more contents. Otherwise with $x > 0$,

$$V_{i,j_1} = \cdots V_{i,j_x} = 1 \tag{23}$$

and

$$V_{i,j_{x+1}} = \frac{R_i - \sum_{c \in C_i} f_c - \sum_{1 \leq m \leq x} f_{j_m}}{f_{j_{x+1}}}, \tag{24}$$

$$V_{j_{x+2}} = \cdots = V_{w_i} = 0 \tag{25}$$

From Equation(21), the first x number of contents in the ordered set can be placed in the cache of node i. The probability of caching $(x+1)$th content depends on the extra room left after allocating space for the first x contents, as calculated in Equation(24).

IV. SIMULATION RESULTS

A. Parameter Settings

In order to keep the optimization problem tractable, we considered a small sized network with 12 CNF routers that totally host 12 contents. We used the Georgia Tech Internetwork Topology Model (GT-ITM) [11] to generate the network topology.

In order to model spatial locality, we assume that requests from an end node are mostly for contents originated from the same stub, with others for remote contents. We define the percentage of requests for same-stub contents to be σ, which is called the locality parameter.

In addition to the proposed Independent Allocation algorithm, we also include four caching schemes for comparison:

- CB-LRU: Content broadcast packets are propagated in the neighborhood to inform the cached copies of contents. Old content or contents are evicted by Least-Recently-Used replacement policy to cache the new one when the storage is limited.
- CB-LPFO: Similar to CB-LRU, but instead of lRU, Least-Popular-First-Out(LPFO) replacement policy is applied. Indicated from the name of LPFO, the least popular content or contents are sacrificed for the new one.

B. Performance Results

1) Impact of Cache Size: In Fig. 3, we compare the average content retrieval latency of CB-LRU, CB-LPFO with our proposed Independent Allocation algorithm. We simulate a wide range of cache sizes in each CNF router, from 10% to 80% of the total size of all contents, and we set the locality parameter σ to be 0.8. All caching schemes with content broadcast

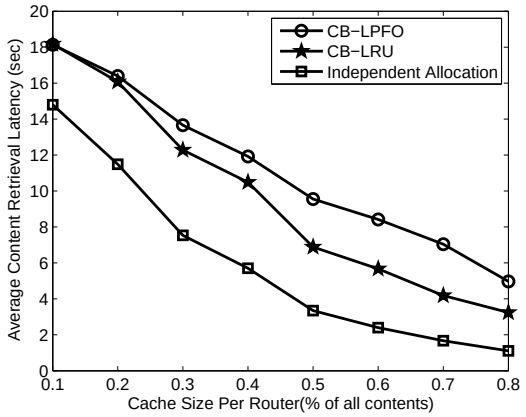

Fig. 3. Average content retrieval latency vs. cache limit on each CNF router.

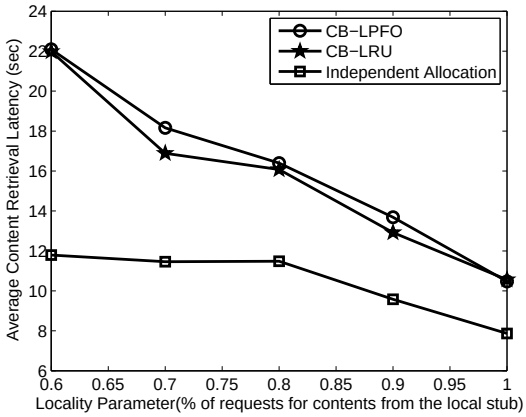

Fig. 4. Average content retrieval latency vs. locality parameter.

provide steady performance improvement as the cache size increases. CB-LRU always shows better performance than CB-LPFO. Since CB-LPFO uses predetermined popularity levels of contents as the replacement index, unable to learn the real content popularity as CB-LRU does by looking at the access rate of cached contents. Independent Allocation algorithm significantly reduces the average content retrieval latency over CB-LPFO and CB-LRU. The improvement increases as the cache size becomes larger. With the largest cache size we have simulated (80% of all contents), the improvement of Independent Allocation algorithm can be as high as 75% over CB-LPFO, and 65% over CB-LRU. And at a medium cache size, such as 20%, the performance improvement can be 30% for both CB-LPFO and CB-LRU.

2) Impact of Locality Parameter: In this set of experiments, we vary the locality parameter σ from 0.6 to 1. The cache size on each router is set to 20%. From Fig.4, we can see that the average content retrieval latency decreases when the locality parameter increases with the three caching schemes. CB-LRU still shows a little better performance than CB-LPFO with different locality parameters. Meanwhile, the Independent Allocation caching scheme significantly reduces the average retrieval latency compared to CB-LPFO and CB-LRU. Larger σ means end users are more likely to request the content originated within the same stub, which makes the retrieval latency smaller due to smaller distance from the original server or in-network cache. The gap is wider between CB-LPFO(and CB-LRU) and the Independent Allocation algorithm when the locality parameter is larger. As more contents transported in the same stub, it is more imperative to make wise decisions as to what contents should be cached to facilitate future retrievals.

V. Conclusions

Cache-and-Forward architecture has been proposed as a solution that leverages the rapidly increasing capacity and dropping costs of data storage to reduce the network traffic and speed up content dissemination. Integrated In-Network Caching framework is one of the key components of the CNF architecture, which incorporates cache into each individual CNF router. In this paper, we further boosted the content retrieval latency performance by Content-Broadcast. Instead of being silent after a content is cached, the CNF router broadcasts this information to its neighbors, enabling optimal and coordinated decision making for caching. We built a rigorous mathematical model on which caching decisions were formulated as an optimization problem. We solved the optimization problem with the Independent Allocation algorithm, which provides an optimal replacement policy when the cache on a CNF router is full. The simulation results show that Independent Allocation algorithm outperforms CB-LRU and CB-LPFO by 65% when the cache size is large. Even with small cache size, the performance gain can reach 30%.

References

[1] D. Raychaudhuri, R. Yates, S. Paul, and J. Kurose, "The Cache-and-Forward Network Architecture for Efficient Mobile Content Delivery Services in the Future Internet," in *Prceedings of ITU-NGN Conference*, 2008.

[2] L. Dong, H. Liu, Y. Zhang, S. Paul, and D. Raychaudhuri, "On the cache-and-forward network architecture," in *Proceedings of the IEEE International Conference on Communications(ICC)*, 2009.

[3] L. Fan, P. Cao, 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.

[4] S. Bhattacharjee, K. L. Calvert, and E. W. Zegura, "Self-organizing wide-area network caches," in *Proceedings of IEEE Infocom*, 1998.

[5] P. Krishnan, D. Raz, and Y. Shavitt, "The cache location problem," *IEEE/ACM Transactions on Networking*, vol. 8, no. 5, pp. 568–582, 2000.

[6] X. Tang and S.T. Chanson, "Coordinated en-route Web caching," *IEEE Transactions on Computers*, vol. 51, no. 6, pp. 595–607, 2002.

[7] D. L. Tennenhouse and D. J. Wetherall, "Towards an active network architecture," *Computer Communication Review*, vol. 26, pp. 5–18, 1996.

[8] Edwin N. Johnson, "A Protocol for Network Level Caching ," M.S. thesis, Massachusetts Institute of Technology, May 1998.

[9] L.H.Lehman, S.J.Garland, and D.L.Tennenhouse, "Active reliable multicast," in *Proceedings of IEEE Infocom*, 1998.

[10] K. Gummadi, R. Dunn, S. Saroiu, S. Gribble, H. Levy, and J. Zahorjan, "Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, 2003.

[11] K. Calvert, M. Doar, and E.W. Zegura, "Modeling Internet Topology," *IEEE Communications Magazine*, vol. 35, no. 6, pp. 160–163, 1997.