Optimized Content Caching and Request Capture in CNF 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*—In order to meet the overwhelming demands of content retrieval for mobile end users, a novel architecture for the next-generation Internet called Cache-and-Forward (CNF) has been proposed to transport content as large packages in a hop by hop manner towards the destination, instead of transporting a stream of small packets along an established TCP/IP connection. In this paper, we investigate how CNF network architecture can be designed for efficient content retrieval for wireless mobile nodes. In particular, we look at Integrated Caching in which we assume each CNF router on the future Internet can cache contents that pass by and reply to content requests with its local copy. We name this content delivery method Cachingn-Capture(CC). We develop a mathematical model for CC to optimize the average content retrieval latency with limited storage on each CNF router. We propose Sequential Reassignment(SR) algorithm to solve the optimization problem.We compare the performance of the derived optimal solutions against our integrated caching and routing heuristics. The results show the Sequential Reassignment algorithm significantly reduces the average content retrieval latency by as high as 70%.**

I. Introduction

In the past few decades, the Internet has enabled a large array of applications, which have profoundly changed the way we interact with the rest of the world. However, as applications become more demanding, and as new technology makes available larger storage, higher bandwidth, as well as diverse means of connecting to the Internet, the current design of the Internet may not be sufficient to address the future needs and opportunities. In response to this challenge, the research community recently initiated an effort aiming the design and evaluation of "clean slate" protocols for the future Internet [1]. One of these clean-slate Internet projects is the "Cache-and-Forward" architecture which proposes the use of hop-by-hop transport along with in-network storage and caching to achieve efficient content delivery to both fixed and mobile end-points [2], [3].

Assumptions of stability and end-to-end connection have traditionally guided the design of TCP/IP protocols, and have led to efficient information transfer and effective recovery strategies during periods of stress. Now, however, this end-to-end strategy is being threatened by a revolution in wireless access technology that alters dramatically the nature of internet traffic, and challenges the basic assumptions upon which its protocols were built. While the end-points of Internet traffic were once stable and predictable, now they are increasingly embodied in wireless devices, whose numbers and information rates are varying and frequent disconnections are common. They have introduced instability to Internet connectivity and made the easy assumptions of end-to-end traffic flow increasingly untenable.

Wireless access rates have increased 50-fold in the last decade, solid-state storage capacities have increased 100-fold, while dropping in cost to $50/GB, and magnetic storage devices have increased 100-fold, while dropping in cost to $0.50/GB. It has become cheap enough to afford putting storage on each individual router, enabling routers to make independent decision whether to cache contents while forwarding them.

Fundamental to CNF architecture are two components: a transport layer service that operates in a hop-by-hop store-and-forward manner, and a content retrieval framework that exploits large, inexpensive storage at each CNF router to cache contents when they are routed through. For mobile nodes, the CNF architecture enables opportunistic push-pull delivery of files, both to and from the wired network. Routing to and from mobile terminals will exploit location information provided by an enhanced name service. Distributed caching of popular content will occur throughout the network, thus making peer-to-peer file sharing a first-class service. We call the CNF caching paradigm as *In-network Caching*. This paper is mainly focused on the design of optimal content dissemination algorithms based on distributed in-network caching.

The content dissemination framework consists of two phases. The first phase is content discovery, which aims at discovering the locations of the requested content based on a distributed directory. The second phase is content retrieval, in which the endpoint acquires the content from the content location. 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 call *Cache-n-Capture*. We note that Cache-n-Capture is not a new approach; in fact, it was discussed in earlier studies, such as en-route caching in [4], [5]. Researchers have been working on coordinated enroute web caching context [5], [6], [7], [8]. However, the performance of en-route web caching depends both on the locations of the caches and how the cache contents are

Digital Object Identifier: 10.4108/ICST.WICON2010.8498
http://dx.doi.org/10.4108/ICST.WICON2010.8498

managed. In Cache-and-Forward networks, cache is integrated to each router. Caching can happen at every enroute router from the server to the mobile end user. Thus it is still a challenging problem whether there is a general principal for each CNF router to follow when a caching decision needed to be made.

In this paper, we formulate a novel mathematical model which takes the content caching and request capture ability of each enroute router into consideration. We propose a distributed caching scheme, which is called Sequential Reassignment. We prove that the algorithm solves the optimization problem and converges to a suboptimal solution. The simulation results further show that the average content retrieval latency performance is much improved under the Sequential Reassignment caching scheme.

The rest of the paper is organized as follows. Section II summarizes related work. We give a brief overview of CNF architecture in Section III. Next, we discuss the content dissemination framework in Section IV. The proposed mathematical model and optimal solution is presented in Section V. The simulation results are shown in Section VI. Finally, we provide concluding remarks in Section VII.

II. Related Work

Disruption/Delay Tolerant Networking (DTN): There are major differences between CNF architecture and DTN architecture [9]. DTN network is driven by *disruption* which implies potentially long periods of disconnection while CNF is driven by a combination of *wireless*, *intermittent connectivity* and *content*. Because of this fundamental difference in the drivers of the design, the architectures have resulted in subtle but fundamental differences.

DTN network is an extension of the TCP/IP network for disconnected environment. As a result, applications interface with DTN network in a manner similar to how they interface with TCP/IP networks. In CNF network, applications interface with the network in a distinctly different way. Specifically, the interface abstraction is that of *content retrieval* as opposed to *conversation*. Therefore, an application would request the network to retrieve a content specified by Content ID (CID), which differs from connecting to a specific node for the purpose of delivering/retrieving information.

DTN routing [10], [11] is again driven by *disconnection*, with the goal of delivering content to a destination which may not be connected. CNF routing has two phases, the first being a *content discovery* phase whereby the network locates/discovers the content requested by the end-user and the second phase is similar to DTN routing. Note that the content discovery phase is built in to the CNF architecture and is not an *overlay* as in DTN architecture.

To summarize, the differences between DTN and CNF architectures stem from the distinctions in the design drivers for the two networks. However, most of the differences are complementary as opposed to conflicting and hence can be incorporated in the next-generation DTN architecture.

Caching: A lot of work has been done in the field of caching. Caching can be implemented in various flavors, namely, hierarchical caching [12], [13], [14], distributed caching with centralized control [15], cooperative caching [16], [17] etc. Much has been done in the placement of caches [18], [19] and cache replacement policies [20], [21], [22].

Most of the work in the literature assumes an overlay of caches on the network. Caches have not been considered as an integral part of the underlying network in the same way routers have been.

The idea of having Internet routers cache passing data has been proposed and discussed in several contexts. For example, in [4], the authors proposed to associate caching with en-route router nodes to speed up object access latency. Several simple association methods were discussed in this paper, namely, caching at every transit node, caching at every gateway node, and independently caching at every router node. The similar idea was also discussed in the context of Active Networks [23]. In[24], a network level caching protocol was proposed to cache individual data packets in the nodes of the network, which can reduce network traffic near the server as well as packet latency. In Active Reliable Multicast [25], routers perform "best-effort" caching of multicast data such that any router on the route of a request can perform retransmission, which can significantly improve the multicast performance. In [26], the authors proposed to cache data on intermediate routers so that the routers can intercept later requests to reduce the server load and the data retrieval time.

In [5], the authors considered the coordinated enroute web caching environment for linear topology. An enroute web caching algorithm was proposed for placing web files at only one node on the path from client to server in the tree network in [6]. In [7], the authors presented a mathematical model to optimally decide where copies of the requested object should be placed for tree networks too. The optimization problems were formulated in single server networks in [5], [6], [7]. In [8], the authors solved the problem of coordinated enroute web caching in multiserver networks, with emergence of various advanced networks that comprise a group of geographically distributed servers. In this paper, we still make the assumption that there is only one original server for each content. However, our mathematical model can be easily extended to multiserver situations. Maximizing cost gain is considered as the optimization objective which is widely used in web services in [5], [6], [7], [8]. However, in Cache-and-Forward networks, quickly satisfying mobile end users' content requests is the main focus. Therefore, we consider a concrete performance objective, called average content retrieval latency throughout the paper.

III. Overview of Cache-and-Forward Architecture

In this section, we present the basic concepts of Cache-and-Forward architecture. We also shed some light on what improvement that CNF achieves over the existing TCP/IP architecture.

Digital Object Identifier: 10.4108/ICST.WICON2010.8498
http://dx.doi.org/10.4108/ICST.WICON2010.8498

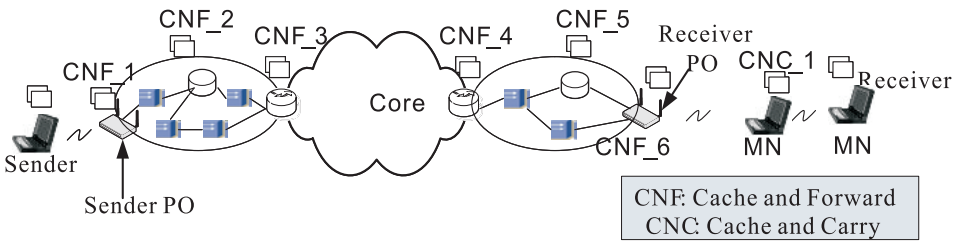

Fig. 1. Cache-and-Forward architecture.

A. Concepts of Cache-and-Forward

The Cache-and-Forward architecture makes the following assumptions:

- The network is hierarchical. As shown in Fig. 1, a very high-bandwidth static core has edge nodes (EN) that connect via a medium-high bandwidth static access network to access nodes (AN) that act as wireless gateways. At the mobile fringe are mobile nodes that connect to the AN via low-medium rate multi-hop wireless links as well as mobile nodes that exploit disconnected high-speed file exchanges. The AN is the aggregation point for the mobile nodes and ad hoc mobile networks, and the EN is the aggregation point for the Access Nodes.
- Transport is provided by Cache-and-Forward(CNF) routers. These CNF routers may appear throughout the network hierarchy, as caching routers or edge nodes in the core, as caching access nodes, or even as caching mobile hosts in the mobile fringe, which are called as Carry-and-Carry(CNC) routers. Both CNF and CNC routers have persistent storage.
- Every mobile node is associated with a set of Post Offices(PO): Typically, access nodes on the wired network will serve as post offices. However, in our design, any CNF node, even a mobile CNF node can be a PO. Each mobile has a list of post office descriptors (POD) that characterizes both the mobile time-varying network connection as well as the properties, such as mobility, of the associated POs. Each mobile node is responsible for updating its post office descriptors. Note that a PO is different from Mobile IP Foreign Agent because the PO is not required to forward data to the mobile; rather the mobile is expected to arrange to pick up any data destined for it from the PO. In addition, unlike Mobile IP, there may be multiple POs corresponding to a mobile node.

B. Advantages of Cache-and-Forward

1) Efficient Multihop Wireless Transmission: Consider an ad hoc wireless network with stationary nodes such that the PHY layer radio connectivity is adequate. Suppose that these nodes are supporting a TCP file transfer over a multihop radio path. In this case, data packets in the forward direction (from sender to receiver) contend for the channel with RTS/CTS at the PHY layer as well with TCP ACK messages in the reverse direction. These contending data packets cause self interference to the multihop route and can disrupt timely control message exchanges. This condition can be perceived as a lost link, triggering inappropriate route repair or route discovery mechanisms, ultimately resulting in transport layer timeouts and dramatic reductions in throughput. This deficiency is in addition to the problems caused by physical layer outages induced by fading on a single link, for which solutions have been developed. Alternatively, hop-by-hop transmission of the file by CNF routers avoids self-interference, since the transmission on any hop does not start until the previous hop is completed. Although this forfeits the potential benefits of pipelining, preliminary experiments[27] indicate that the reduction in self-interference more than compensates.

2) Facilitating cache-and-carry to increase capacity in mobile scenarios: Cache-and-Forward allows for a seamless unified routing solution for wired and wireless networks. In this case, a route could be a sequence of hosts capable of sustaining a real-time connection or a sequence of hosts(CNC routers) physically carrying the data, or even some combination of these approaches. The potential diversity of routes is increased because an end-to-end real-time connection is no longer mandated.

3) Making content sharing a first class service: Cache-and-forward can provide benefits in the wired Internet where peer-to-peer (P2P) traffic has become widespread. Since P2P data transfer is not a ”service” offered by the Internet, several independent applications with very different architecture and protocols have been developed to accomplish what is essentially the same result. Such peer-to-peer file transfers have become so common that it is worth having a common service from the network to meet these needs. This is analogous to TCP, without which each application that requires reliable transport of packets would have had to develop its own reliable transport protocol. Just as TCP offered a ”reliable byte stream” service to the hosts connected to the Internet, the Cache-and-Forward architecture can provide an ”efficient file transport service” between hosts on the next generation network.

In such an architecture, caching of popular files becomes a natural component of the network layer. Multiple copies of any large content file may be stored in caches to maximize the probability of timely delivery when the location of the

Digital Object Identifier: 10.4108/ICST.WICON2010.8498
http://dx.doi.org/10.4108/ICST.WICON2010.8498

Algorithm 1 The routine on an endpoint.

$contentLocationList \leftarrow$ **FindLoc**(CID)
$bestLocation \leftarrow$
CalculteBestLocation($contentLocationList$)
$content \leftarrow$ **RetrieveContent**(CID, $bestLocation$)

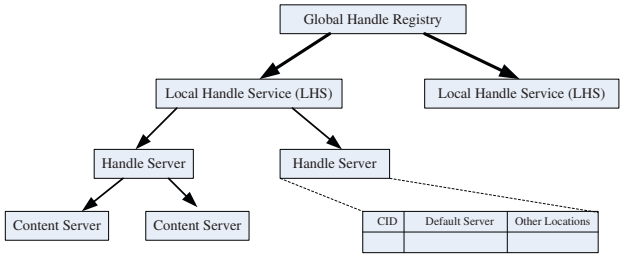

Fig. 2. CNRS implementation using handle system.

recipient is not certain.

IV. Content Dissemination Framework in a CNF Network

The content dissemination framework consists of two phases. The first phase is *content discovery*, in which a requestor discovers the location(s) of the desired content files. Following that, the second phase is *content retrieval*, in which the endpoint sends a request towards the hosting server, and the requested content will be returned to the requestor as the outcome. This framework is summarized in Algorithm 1. In the rest of this section, we discuss these two phases in detail.

A. Content Discovery Through Content Name Resolution Service (CNRS)

To make the network centered around content delivery, we introduce the notion of persistent, globally unique content identifiers, referred to as *CID*'s. CID's must be location-independent: A content file stored in multiple locations within the CNF network will be referred to by the same CID. In today's Internet, content is identified by a URL whose prefix consists of a string denoting the location of the content, which is not location-independent, and thus is not a good candidate for CID. One possible candidate for CID is the notion of a handle as in the Handle System [28]. In the Handle system, upon creation, a content file is assigned to a unique CID, and the CID stays the same throughout its lifetime, even though the content may be relocated or replicated to multiple locations. An important service a CNF network provides is for endpoints to retrieve specific content. Here, we assume that endpoints can obtain the content's CID before hand through a search engine-like service.

After obtaining the CID, an endpoint then needs to find the locations of the content through the Content Name Resolution Server (CNRS), which maps a CID to a list of hosts (and/or their mirrors) that have a copy of the content. A possible implementation of CNRS would be through the handle system that consists of a global handle registry, local handle services, and handle servers that form a hierarchical structure, as shown in Fig.2. The global handle registry can map a CID to the corresponding local handle services providing links to handle servers, which in turn store the locations of the content.

In such a system, a `Content Name Resolution` request from the requestor endpoint is first routed to the requestor's local handle service node. If the requested CID is not found on the local handler service node, the request will be routed to the global handle registry, which then forwards the request to the destination local handle service node. The destination local handle service will then consult its handle server and return a list of servers/hosts of the content to the requestor. The location list consists of two parts: the default server, and the other servers (or mirrors). The list will be routed back to the requesting endpoint using IP protocols.

From the list of content sources, an endpoint can select the most suitable source node to send a `Content Retrieval` request to. An endpoint can make this selection based on various criteria, and one likely candidate is the closest source node. The request is routed towards the chosen content source using IP protocols.

In this paper, we consider every content only has one original server, the mapping between a CID and its server is maintained by the CNRS service.

Algorithm 2 Cache-n-Capture routine on a CNF router.

loop
  /* Capture */
  if The CNF just received a ContentRetrieval request originated from node S for content CID **then**
    $cached \leftarrow$ **CheckCache**(CID)
    if $cached$ =**true then**
      Send(CID, S, $content$)
    else
      Forward(req, $nextHop$)
    end if
  end if

  /* Cache */
  if The CNF just received a ContentReply destined to node S with content CID **then**
    $toCache \leftarrow$ **CheckCachingCriteria**(CID)
    if $toCache$ =**true then**
      Cache(CID)
    else
      Forward($reply$, $nextHop$)
    end if
  end if
end loop

B. Content Retrieval Method: Cache-n-Capture (CC)

In Cache-n-Capture (CC), both content caching and access are passive. Here, a CNF router only caches those content files that are routed through it (i.e., *caching*), and a CNF router only helps return the cached copy to those requests that are routed through it (i.e., *capture*).

First, let us look at the capture part of CC. A CNF router, after receiving a `Content Retrieval` request, first checks whether the content is cached locally. If the CNF router has a copy of the requested content, it stops forwarding the request

Digital Object Identifier: 10.4108/ICST.WICON2010.8498
http://dx.doi.org/10.4108/ICST.WICON2010.8498

to the destination node, but instead, it returns the local copy to the requestor hop by hop.

Next, let us look at the caching part of CC. When a content file is routed through a CNF router towards the requesting endpoint, the CNF router can choose to cache the content based on some criteria. Below are some example criteria:

- *Popular content.* Content popularity can be indicated by the content sources.
- *Interest level.* A CNF router can maintain a list of content files to which the number of requests has exceeded a threshold, and chooses to cache these files when they pass by.
- *Source specification.* Sometimes the content source can specify whether and how the cache should be cached, e.g. caching at every hop.

Another issue that is worth noting is the cache replacement policy. When the cache of a CNF router becomes full, it can evict a victim content to accommodate the new content. The victim can be selected based on a range of policies, including First In First Out(FIFO), Least Recently Accessed First(LRU).

The algorithm of CC is summarized in Algorithm 2.

V. The CC Optimization Framework

In this study, we formulated the Caching-n-Capture strategy as an optimization problem with the objective of minimizing average content retrieval latency. Since the storage on each CNF router in the static access network can not be infinitely large to cache all contents, the problem is to answer what is the optimal set of contents that should be cached in each CNF router with the limited storage.

A. System Model

To formulate the optimization problem, we assume that the popularity distribution of the contents is known a priori, which follows the MZipf distribution, which defines the probability of retrieving the i-th content out of F available contents as

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

and

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

where α 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.

Since in CNF architecture, the access node is aggregation point for the mobile nodes and ad hoc mobile networks, it represents the mobile nodes to send out content retrieval requests. Thus we model the static core and access network to be as an undirected graph $G = (V, E)$, where a vertex in V represents a node, and an edge in E represents a network link. An access node may request the same content more than once, if there are different mobile nodes connecting to it which have the same interest to that content. We also assume each node in the network is a CNF router, which has persistent but limited storage.

Before we introduce our models, we first summarize the parameters and variables used in the model. The following parameters are defined in the formulation:

- Nodes: $1, 2, ...N$;
- Contents: $1, 2, ...F$;
- Link bandwidth: B;
- Request packet size: Q;
- Size of content i : f_i;
- Processing delay at each node: D_p;
- Per-hop request transmission delay: $D_q = Q/B$;
- Per-hop content transmission delay: $d_i = f_i/B$;
- Total delay of requesting content i over a hop: $D_i = D_q + D_p + d_i$;

In addition, the following variables are defined:

- $P_{i,j}$, the probability for node i to request content j;
- $V_{i,j}$, the probability for node i to cache content j;
- R_i, the storage limit of node i;

B. Problem Formulation

To formulate CC, we need to define the following additional variables:

- S_j, the original server of content j;
- $H_{a,b}$, the hop count of the shortest path from node a to node b;
- $C_h^{a \to b}$, the h-th node on the routing path from node a to b (assuming a is the 0-th node on the path);

The objective function is to minimize the average latency, which can be realized through minimizing the sum of the latencies for all requests:

$$min \sum_{i=1}^{N} \sum_{j=1}^{F} P_{i,j} \cdot \Big(\sum_{h=1}^{H_{i,S_j}} \big(h \cdot D_j \cdot V_{C_h^{i \to S_j},j} \cdot \prod_{k=0}^{h-1} (1 - V_{C_k^{i \to S_j},j})\big)\Big) \tag{3}$$

In Equation 3, the product $\prod_{k=0}^{h-1}(1 - V_{C_k^{i \to S_j},j})$ denotes the probability that the query from node i for content j is not captured by the first $h-1$ nodes on the routing path to the server. The h-th node on the path (excluding the requester) is denoted by $C_h^{i \to S_j}$, and the probability that it holds the requested content is $V_{C_h^{i \to S_j},j}$. Finally, the product, $h \cdot D_j$, is the latency of retrieving content j from this node.

We have the following constraints:

$$\sum_{j=1}^{F} V_{i,j} \cdot f_j \leq R_i \tag{4}$$

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

$$0 \leq V_{i,j} \leq 1 \tag{6}$$

The first constraint is to make sure that the total size of the contents cached on each node will not exceed its storage limit.

Digital Object Identifier: 10.4108/ICST.WICON2010.8498
http://dx.doi.org/10.4108/ICST.WICON2010.8498

The second constraint states that the probability that content j resides in its server should be 1. The third constraint is to ensure the probability for node i to cache content j is between 0 and 1.

Algorithm 3 $A(i)$ (Reassignment at node i)

Step 1: Calculate $c_{i,j}$

for $j = 1$ to F **do**

if $i \neq S_j$ **then**

$$c_{i,j} = \sum_{\substack{i \in (k,S_j) \\ i \neq k}} P_{k,j}(h_{k,i,S_j} - 1)D_j \prod_{\ell=0}^{h_{k,i,S_j}-1} \left(1 - V_{C_\ell^{k \to S_j},j}\right) - \sum_{i \in (k,S_j)} P_{k,j} \left(\sum_{h=h_{k,i,S_j}+1}^{H_{k,S_j}} (h-1) D_j V_{C_h^{k \to S_j},j} \prod_{\substack{\ell=0 \\ \ell \neq h_{k,i,S_j}}}^{h-1} \left(1 - V_{C_\ell^{k \to S_j},j}\right) \right) \tag{7}$$

end if
end for

Step 2: Sort $c_{i,j}/f_j$
such that

$$c_{i,j_1}/f_{j_1} \leq c_{i,j_2}/f_{j_2} \leq \cdots \leq c_{i,j_{t_i}}/f_{j_{t_i}} \tag{8}$$

where t_i denotes the size of the set of $j \in \{1, 2, \ldots, F\}$ such that $i \neq S_j$;

Step 3: Find extra storage space of node i
Find $0 \leq d \leq t_i$ such that

$$\sum_{\substack{1 \leq m' \leq F \\ i = S_{m'}}} f_{m'} + \sum_{1 \leq m \leq d} f_{j_m} \leq R_i \tag{9}$$

and for any $n > d$,

$$\sum_{\substack{1 \leq m' \leq F \\ i = S_{m'}}} f_{m'} + \sum_{1 \leq m \leq n} f_{j_m} > R_i \tag{10}$$

Step 4: Reassign the values of $V_{i,j}$
Set $V_{i,j} = 1$ for $S_j = i$.
if $d = 0$ **then**
Set $V_{i,j} = 0$, if $S_j \neq i$;
else
Set

$$V_{j_1} = \cdots V_{j_d} = 1 \tag{11}$$

and

$$V_{i,j_{d+1}} = \frac{R_i - \sum_{\substack{1 \leq m' \leq F \\ i = S_{m'}}} f_{m'} - \sum_{1 \leq m \leq d} f_{j_m}}{f_{j_{d+1}}}, \tag{12}$$

$$V_{j_{d+2}} = \cdots = V_{t_i} = 0 \tag{13}$$

end if

C. Sequential Reassignment Algorithm

In this subsection we give a distributed algorithm that can obtain a suboptimal solution to (3), which is called Sequential Reassignment Algorithm. As suggested by its name, the atomic operation of the reassignment algorithm is **reassignment at node** i which is denoted by $A(i)$. The execution of the reassignment algorithm simply consists of each node randomly or periodically executing $A(i)$, whose details are given in Algorithm 3. $A(i)$ adjusts $V_{i,j}$ for $j = 1, 2, \ldots, F$ for local optimality. When $A(i)$ is executed, $V_{i',j}$, $i' \neq i$, $j = 1, 2, \ldots, F$, are assumed to be feasible (meeting the respective constraints) and fixed. Though the algorithm we present here implicitly assumes each node i somehow knows $V_{i',j}$ for all j and i' such that $i \in (i', S_j)$ (the shortest path from node i' to the original server of content j), in practice, these quantities can be piggybacked by requests from i' or be estimated at node i by observing request patterns of node i'. In Algorithm 3, $c_{i,j}$ is the derivative of the objective function corresponding to $v_{i,j}$. The morale of the reassignment algorithm comes from the following observations.

Proposition 1: $c_{i,j} \leq 0$.

Proof: As given in the assumption for $A(i)$, when $V_{i',j}$ ($\forall j$ and $\forall i' \neq i$) are known as fixed feasible values at node i, (3) can be rewritten as

$$\text{minimize} \quad \sum_{j=1}^{F} c_{i,j} V_{i,j} + \text{some constants}, \tag{14}$$

$$\text{subject to} \quad \sum_{j=1}^{F} V_{i,j} f_j \leq R_i, \tag{15}$$

$$0 \leq V_{i,j} \leq 1, \quad \text{if } S_j \neq i, \tag{16}$$

$$V_{i,j} = 1, \quad \text{if } S_j = i. \tag{17}$$

after some algebra. Though (7) that gives the formula of $c_{i,j}$ 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 (14), we find this is possible only when $c_{i,j} \leq 0$. ■

Proposition 2: $A(i)$ solves the optimization problem in (14).

Proof: If $\sum_{j=1}^{F} f_j \leq R_i$, then the optimal solution is trivially $V_{i,j} = 1, \forall j$, which is accomplished by $A(i)$. Now assume $\sum_{j=1}^{F} f_j > R_i$. Form the partial Lagrangian of (14)

$$\text{minimize} \quad \mathcal{L} = \sum_{j=1}^{F} c_{i,j} V_{i,j} + \mu \left(\sum_{j=1}^{F} f_j V_{i,j} - R_i \right) \tag{18}$$

$$= \sum_{j=1}^{F} (c_{i,j} + \mu f_j) V_{i,j} - \mu R_i \tag{19}$$

$$= \sum_{j=1}^{F} f_j (c_{i,j}/f_j + \mu) V_{i,j} - \mu R_i \tag{20}$$

Digital Object Identifier: 10.4108/ICST.WICON2010.8498
http://dx.doi.org/10.4108/ICST.WICON2010.8498

subject to (16) and (17), with $\mu \geq 0$ be the dual variable. Suppose the optimal dual variable μ satisfies $\mu = 0$. Since we know $c_{i,j} \leq 0$ by Proposition 1, 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 $\mu > 0$ and, by complimentary slackness, (15) is tight. The optimal solution is hence obtained by setting $V_{i,j} = 1$ if $S_j = i$ and setting

$$V_{i,j} = \begin{cases} 1, & c_{i,j}/f_j + \mu < 0, \\ 0, & c_{i,j}/f_j + \mu > 0, \\ \text{any feasible value}, & c_{i,j}/f_j = 0, \end{cases} \tag{21}$$

for those j if $S_j \neq i$, such that $\mu > 0$ and (15) is tight. Readers can check that one such solution is constructed by $A(i)$ if we let $\mu = -c_{i,d+2}/f_{d+2}$. ■

Corollary 1: The reassignment algorithm converges to a suboptimal solution.

Proof: By Proposition 2, each execution of $A(i)$, in any order, can only result in a smaller and better optimal objective value in (3). Since the objective value is lower bounded by 0. The reassignment algorithm converges to a suboptimal solution. ■

Corollary 2: If the reassignment algorithm discovers a optimal solution, every further execution of $A(i)$ can only result in an optimal solution.

Proof: Obvious. ■

VI. Simulation Results

A. Parameter Settings

1) Network Topology: In order to keep the optimization problem tractable, we considered a network with 12 CNF routers that totally host 12 contents. We used the Georgia Tech Internetwork Topology Model (GT-ITM) [29] [30] to generate the network topology. The network consists of one transit network and two access networks. Within a stub, the stub nodes represent access nodes for mobile end users in local area networks (LANs) (each of these stub nodes represents one LAN), and as a result, these stub nodes generate user requests. Each CNF router can cache contents that pass by.

2) Normalized Content Request Probability: In order to solve the objective functions, the key is to calculate $P_{i,j}$, the unified probability for node i to request content j. In order to model spatial locality, we assume that requests from an end node are most for contents originated from the same stub, others are for remote contents. We define this percentage to be σ, which is called locality parameter.

Calculating the variable $P_{i,j}$ involves two steps. First, from Equation 1, we can derive the probability of retrieving an content at a certain rank out of a total of F contents. Second, we should take the locality property of content retrieval into consideration. Thus, we have:

$$P_{i,j} = \begin{cases} 0, & \text{if } i \text{ is the server of content } j \\ \sigma \times \frac{1}{N} \times \frac{Pr(j)}{\sum^{h \in SS_i} Pr(h)}, & \text{if } j \in SS_i \\ (1-\sigma) \times \frac{1}{N} \times \frac{Pr(j)}{\sum^{h \in DS_i} Pr(h)}, & \text{if } j \in DS_i \\ \frac{1}{N} \times \frac{Pr(j)}{\sum^{h \in DS_i} Pr(h)}, & \text{if } j \in DS_i \text{ and } SS_i \text{ is empty} \\ \frac{1}{N} \times \frac{Pr(j)}{\sum^{h \in SS_i} Pr(h)}, & \text{if } j \in SS_i \text{ and } DS_i \text{ is empty} \end{cases} \tag{22}$$

$$SS_i = \{\text{contents hosted in the same stub as node } i\}$$
$$DS_i = \{\text{contents hosted in different stubs}\}$$

B. Caching Schemes Evaluated

In addition to the proposed Sequential Reassignment caching algorithm, we included four caching schemes:

- LPFO-every: Contents are cached on every enroute CNF router from the original content server to the requester if there is extra storage. Otherwise, Least-Popular-First-Out(LPFO) replacement policy is applied, which means a CNF router always evicts the least popular content from its cache to provide room for the new one.
- LPFO-even: Contents are cached on enroute CNF routers which are even hops away from the original server. In any case, the requester always caches the content. LPFO replacement policy is also used.
- LRU-every: It is similar to LPFO-every. Instead of Least-Popular-First-Out, LRU replacement policy is employed. If thre is no enough free space, the CNF router purges one least recently accessed content to make room for the new one.
- LRU-even: It only varies from LPFO-even in the replacement policy. LRU is used instead.

C. Performance Results

1) Impact of Cache Size: In Fig. 3, we compare the average content retrieval latency of the four caching schemes. The simulations were made across a wide range of cache sizes, from 10% to 80%, and we set the locality parameter σ to be 0.8. All caching schemes provide steady performance improvement as the cache size on each CNF router increases. The curves of LPFO-every and LPFO-even overlap. With LPFO replacement policy, caching contents on every or every other enroute CNF router does not make much difference in the resulting performance. The two schemes with LRU replacement policy achieves better average retrieval latency than those two schemes with LPFO. Compared to LPFO, LRU running on each intermediate CNF router is more capable of learning the real content popularity by looking at the access rate of cached contents. The caching scheme LRU-every outperforms other three schemes, thus in Fig.4 we only compared the proposed Sequential Reassignment algorithm with LRU-every.

Digital Object Identifier: 10.4108/ICST.WICON2010.8498
http://dx.doi.org/10.4108/ICST.WICON2010.8498

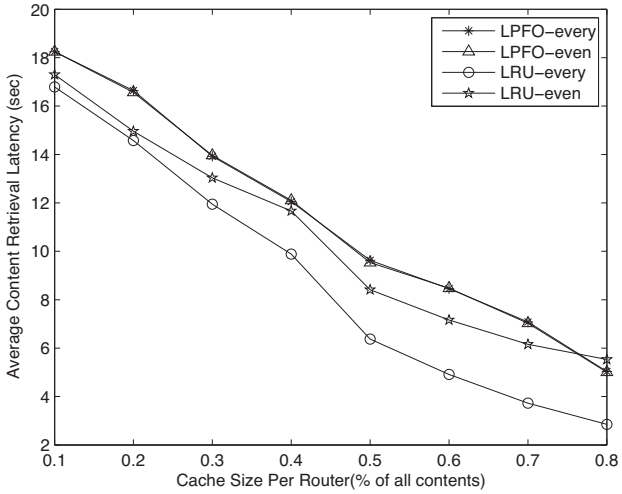

Fig. 3. Average content retrieval latency with four caching schemes

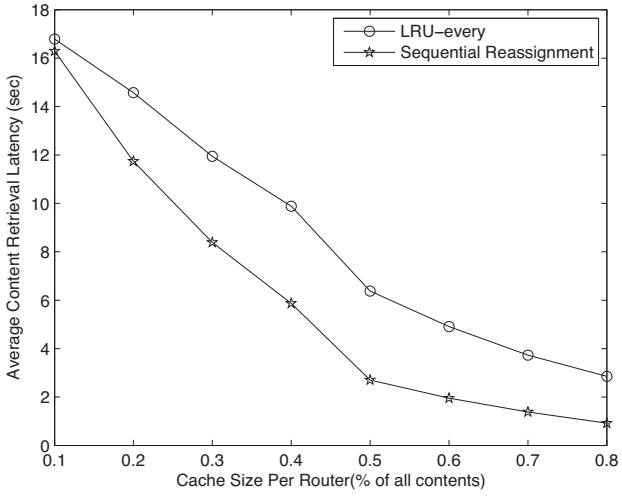

Fig. 4. Average content retrieval latency with Sequential Reassignment

From Fig.4, we can see that Sequential Reassignment algorithm significantly reduces the average content retrieval latency. The relative improvement over LRU-every increases with the cache size on each CNF router. The performance gain can be as high as 70% when the cache size is 80%. At a reasonable cache size, such as 20% or 30%, the performance improvement can reach 20% and 30% respectively.

2) Impact of Locality Parameter: In this set of experiments, we varies the locality parameter σ from 0.6 to 1. The cache size on each CNF router is set to 20%. From Fig.5, we can see that the average content retrieval latency decreases with the locality parameter for both LRU-every and Sequential Reassignment. Larger σ means mobile end users tend to request the contents originated from the same stub, which makes the retrieval latency small due to small distance from the original server or enroute cache. The gap is more widened between LRU-every and Sequential Reassignment when the

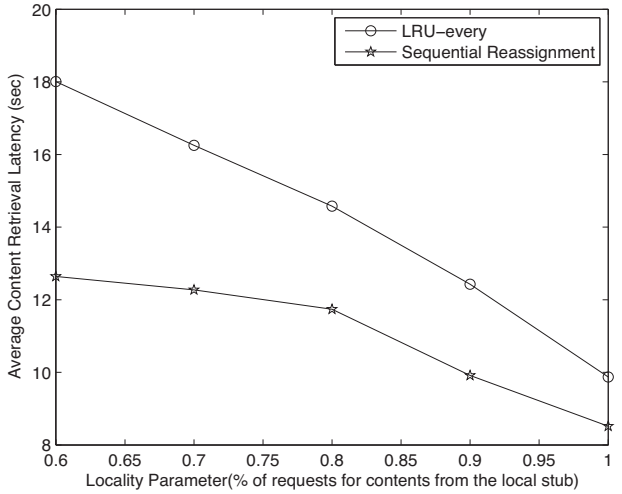

Fig. 5. Average content retrieval latency vs. locality parameter

locality parameter is smaller. Sequential Reassignment shows outstanding efficiency in guiding enroute CNF routers to cache proper contents.

VII. Concluding Remarks

The Cache-and-Forward (CNF) Internet architecture is a significant departure from TCP/IP based Internet architecture in that it opportunistically transports named contents in "packages" in a hop-by-hop manner. CNF is designed to solve the problem of content dissemination and content retrieval in future Internet with a significant number of intermittently connected mobile endpoints. Such an architecture is made feasible by improving cost-performance of storage and computation at routers.

The CNF network has two key components when it comes to content dissemination and content retrieval: (1) in-network caching and (2) content-enhanced routing. Caching in the CNF network is distinctively different from caching in today's Internet because caching is built into the very fabric of the CNF network by allowing each individual CNF router to cache rather than building caching as an overlay infrastructure on top of the core TCP/IP network. This is instrumental in keeping content close to the requester, no matter whether the content is globally or locally popular.

Based on this architecture, we presented the content dissemination framework, which consists of two phases: content discovery and content retrieval. We focused on the content retrieval method: Cache-n-Capture. An intermediate CNF router can cache those content files that are routed through it and helps return the cached copy to those requests that pass by.

We developed a mathematical model for CC to solve the problem how a CNF router independently decides which contents should be cached. We proposed the Sequential Reassignment algorithm to solve the optimal problem. The simulation results show that SR outperforms the simple enroute caching with LPFO and LRU replacement policies by as high as 70%

Digital Object Identifier: 10.4108/ICST.WICON2010.8498
http://dx.doi.org/10.4108/ICST.WICON2010.8498

when the cache size is large. Even with small cache size, the performance gain can achieve 30%.

REFERENCES

[1] "NSF NeTS FIND Initiative," http://www.nets-find.net/.

[2] 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 *ITU-NGN conference*, 2008.

[3] 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.

[4] S. Bhattacharjee, K. L. Calvert, and E. W. Zegura, "Self-organizing wide-area network caches," in *Proceedings of IEEE Infocom'98*, vol. 2, 1998.

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

[6] A. Jiang and J. Bruck, "Optimal content placement for en-route web caching," in *Proc. the 2nd IEEE International Symposium on Network Computing and Applications*, 2003, pp. 9–16.

[7] K. Li, H.Shen, F. Chin, and S.Zheng, "Optimal methods for coordinated enroute web caching for tree networks," *ACM Transactions on Internet Technology*.

[8] H. Shen and S. Xu, "Coordinated en-route web caching in multiserver networks," *IEEE Transactions on Computers*, vol. 58, no. 5, pp. 605–619, 2009.

[9] K. Fall, "A delay tolerant network architecture for challenged internets," in *Proceedings of SIGCOMM*, 2003.

[10] M. Mirco, H. Stephen, and M. Cecilia, "Adaptive routing for intermittently connected mobile ad hoc networks," in *Proceedings of the IEEE 6th International Symposium on a World of Wireless, Mobile, and Multimedia Networks (WoWMoM 2005). Taormina, Italy.*, 2005.

[11] A. Lindgren, A. Doria, and O. Scheln, "Probabilistic routing in intermittently connected networks," in *Proceedings of the Fourth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2003.

[12] A. Chankhunthod, P. B. Danzig, C.Neerdaels, M. F. Schwartz, and K. J. Worrell, "A hierarchical internet object cache," in *USENIX Annual Technical Conference*, 1996, pp. 153–164.

[13] H. Che, Y. Tung, and Z. Wang, "Hierarchical web caching systems: Modeling, design and experimental results," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, 2002.

[14] P. Rodriguez, C. Spanner, and E. Biersack, "Analysis of web caching architectures: Hierarchical and distributed caching," *IEEE/ACM Trans. on Networking*, vol. 9, no. 4, pp. 404–418, 2001.

[15] S. Paul and Z. Fei, "Distributed caching with centralized control," *Computer Communications*, vol. 24, no. 2, pp. 256–268, 2001.

[16] M. Dahlin, R. Wang, T. E. Anderson, and D. A. Patterson, "Cooperative caching: Using remote client memory to improve file system performance," in *Operating Systems Design and Implementation*, 1994.

[17] L. Yin and G. Cao, "Supporting co-operative caching in ad hoc networks," in *Proc. of IEEE INFOCOM*, 2004.

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

[19] E. Cohen and S. Shenker, "Replication strategies in unstructured peer-to-peer networks," in *Proc. of the ACM SIGCOMM*, 2002.

[20] A. Wierzbicki, N. Leibowitz, M. Ripeanu, and R. Wozniak, "Cache replacement policies revisited: the case of p2p traffic," in *IEEE International Symposium on Cluster Computing and the Grid*, 2004.

[21] J. Xu, Q. Hu, W. Lee, and D. Lee, "Performance evaluation of an optimal cache replacement policy for wireless data dissemination under cache consistency," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 1, pp. 125–139, 2004.

[22] F. Benevenuto, F. Duarte, V. Almeida, and J. Almeida, "Web cache replacement policies: properties, limitations and implications," in *Proceedings of the Third Latin American Web Congress*, 2005.

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

[24] E. N. Johnson, "A protocol for network level caching," Master's thesis, Massachusetts Institute of Technology, May 1998.

[25] L.H.Lehman, S.J.Garland, and D.L.Tennenhouse, "Active reliable multicast," in *Proceedings of Infocomm 98*, vol. 2, 1998.

[26] U. Legedza, D. Wetherall, and J. Guttag, "Improving the performance of distributed applications using active networks," in *Proceedings of IEEE Infocom 98*, 1998.

[27] H. Liu, Y. Zhang, S. Paul, and D. Raychaudhuri, "Performance evaluation of the "cache-and-forward (cnf)" network for mobile content delivery services," in *IEEE International Conference on Communications(ICC) Workshops*, 2009.

[28] S. Sun, L. Lannom, and B. Boesch, "RFC 3650:handle system overview," November 2003.

[29] K. Calvert, M. Doar, and E. Zegura, "Modeling internet topology," *IEEE Communications Magazine*, vol. 35, no. 6, pp. 160–163, 1997.

[30] E. Zegura, K. Calvert, and S. Bhattacharje, "How to model an internetwork," in *Proceedings of IEEE Infocom*, 1996.

Digital Object Identifier: 10.4108/ICST.WICON2010.8498
http://dx.doi.org/10.4108/ICST.WICON2010.8498