Performance Comparison of Content-oriented Networking Alternatives: A Hierarchical Tree versus A Flat Distributed Hash Table

Jaeyoung Choi, Jinyoung Han, Eunsang Cho, Taekyoung Kwon, Hyunchul Kim, Yanghee Choi
School of Computer Science and Engineering
Seoul National University, Seoul, Korea
Email: {jychoi, jyhan, escho}@mmlab.snu.ac.kr, tkkwon@snu.ac.kr, hkim@mmlab.snu.ac.kr, yhchoi@snu.ac.kr

***Abstract*—The internet was designed with host-oriented networking applications such as file transfer and remote login. However, recent internet statistics show that content-oriented traffic (e.g. web pages, multimedia clips) becomes more and more dominant. Even though content-oriented networking has received increasing attention, there have been few comprehensive and quantitative studies on how to realize a content-oriented networking framework. In this paper, we focus on the operational issues of the new networking framework: (i) how to locate contents, (ii) how to deliver contents, and (iii) how to cache contents. There are two major infrastructure alternatives in substantiating these mechanisms: a tree and a distributed hash table (DHT). We carry out comprehensive simulation experiments to compare these alternatives in terms of content transfer latency, cache effectiveness, and failure resilience.**

I. Introduction

The Internet has evolved from a small-scale academic testbed into a crucial social infrastructure. For 40 years since its inception, the Internet has tried to accommodate new and unanticipated requirements. However, its rigid design based on TCP/IP, internet service providers' exclusive control of networks and so on have ossified the Internet [6]. One of the Internet design principles that hinders the current Internet evolution is the host-oriented communications. That is, the early stage Internet applications, such as remote login, file transfer, and e-mail, focused strictly on host-to-host communications, where an end user explicitly directs his/her host to communicate with another host.

Over the past several years, however, the vast majority of Internet usage has become data retrieval and service access [23], [3], [16] (i.e., *data-oriented* usage), where an end user cares about contents[1] but is oblivious to or less aware of the host (or its location) [16]. For example, (i) many people often use search engines like Google from which they jump to the data or service page irrespective of the location in which the desired data/service resides, (ii) web intermediaries such as web caches and content delivery networks (CDNs) [18] transparently redirect web clients to a nearby copy of the requested data without contacting the original Web server, and (iii) peer-to-peer (P2P) applications enable users to search and retrieve the data without making them know the location or identity of the host.

The common viable contribution of search engines, web caching, CDNs, and P2P applications is that they all seek to support data-oriented applications. They improve the data availability, ease-of-use, performance, and scalability of applications by overcoming the limitations of the host-oriented principles of the legacy Internet by manipulating DNS naming and/or name resolution. In other words, they decouple content files from their locations (or hosts) using various techniques such as caching, replication with transparent request redirection, maintenance of a searching infrastructure, etc. [16].

Recently, lessons from these several application/service-dependent ad-hoc solutions, have motivated "clean-slate" efforts [1], [16], [10], termed data-oriented or content-centric networking, which strive to redesign the host-oriented Internet to accommodate more and more dominant data-oriented Internet usage at the architectural level. The essence of content-oriented networking lies in decoupling contents from the service host/location not at application levels, but at networking levels. That is, content-oriented networking shifts the focus from transmitting data between two hosts (or their locations) to delivering data via a content identifier[2]. The content-oriented networking architecture is expected to (i) free application/service developers from re-inventing the necessary mechanisms, and (ii) provide scalable and efficient delivery of the requested contents. However, there have been few studies that how to substantiate the content-oriented networking infrastructure in terms of performance metrics.

In order to address the operational issues that are missing in the recent architectural studies on content-oriented networking in the literature, this paper explores two alternatives for content-oriented network infrastructures: a hierarchical tree and a flat distributed hash table (DHT). Whether we choose a tree or a DHT for the infrastructure affects almost every aspect of network operations: name resolution, content delivery, caching, and so on. We will compare these two alternatives in terms of scalability, robustness, transfer latency thoroughly.

[1]In this paper, we use "data" and "contents" almost interchangeably. In the literature, "contents" often have more semantics than "data".

[2]As an example, the Data-Oriented Network Architecture (DONA) [16] replaces endpoint-based DNS by a new name resolution mechanism using flat, certifying names and name-based anycast primitives above the IP layer.

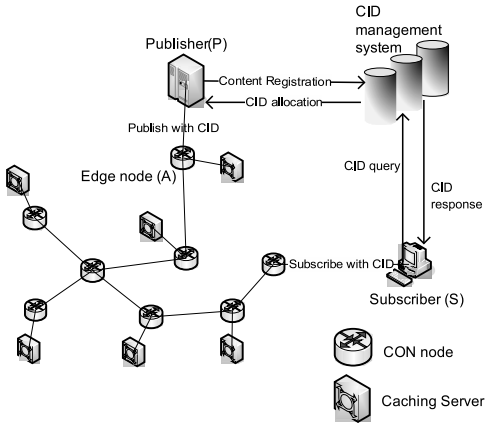

Fig. 1. An illustration of a content-oriented network

The remainder of this paper is organized as follows. Section II introduces content-oriented networking as well as its architectural design criteria and alternatives. Section III presents our experimental settings followed by the simulation results. We discuss related work in Section IV. Section V gives concluding remarks.

II. Content-oriented Networking

The *content-oriented network* is a network which operates under the data-oriented paradigm[3]. As mentioned before, the host-oriented naming in the legacy internet is shifted to the data-oriented one in content-oriented network. Thus, the content-oriented networking framework should have four building blocks: (i) how to identify the contents "in" the network (which is naming), (ii) how to locate contents (which is name resolution), (iii) how to deliver the contents, and (iv) how to cache contents. In this paper, we mainly focus on the last three issues.

The content-oriented networking framework necessitates a new naming scheme for contents. We assume the each content file has its own unique *Content Identifier* (CID). As discussed in [16], [1], once decided, the CID should not be changed (which is persistence), and the CID should be able to authenticate the content publisher (which is authenticity) for the content-oriented network. For our evaluation, we use a flat CID (same as DONA) in order to concentrate on the performance of content-oriented networking alternatives. To simplify the CON architecture to be evaluated, we assume that a CID management system (e.g. [25], [12]) has already been established to assign or to find out CIDs for content files.

Figure 1 illustrates a content-oriented network (CON), which consists of three kinds of entities: (i) a *CON node*[4], (ii) a *caching server*[5], and (iii) an end host which publishes or subscribes contents. *CON nodes* (or shortly, nodes) form an overlay network[6] for end hosts. A *caching server* selectively (and often temporarily) stores content files which are forwarded via its associated CON node. If a cache hit happens, the associated CON node directly forwards the content file on behalf of the publisher.

End hosts perform two kinds of functions: *publishing* and *subscribing*. A *publisher* has contents to be distributed, and a *subscriber* retrieves contents from the content-oriented network. We denote a CON node which serves an end-host directly by an *edge node*. For instance, in Figure 1, node A is the edge node of publisher P. An end-host knows the location of its edge node by some measures as similar to local DNS server configuration. If in-network caching is enabled, not only the publisher, but also a caching server can store contents. We collectively call them *data holders*.

A. Name Resolution

There are mainly two name resolution mechanisms in the context of content-oriented networking: *lookup-by-name* and *route-by-name*. When the CID is given, "lookup-by-name" refers to a process to find out an IP address of the publisher (which is similar to DNS in the host-oriented Internet). Meanwhile, "route-by-name" means how to locate a corresponding content file when the CID is given (e.g. [16], [13]). We will now elaborate on the latter, which is appropriate in content-oriented networking since it decouples the content and its location of the content. For example, the decoupled principle naturally supports anycasting from a subscriber to one of data holders.

Due to the proliferation of large scale distributed systems, there are a number of studies on the publish/subscribe communication paradigm (e.g. [11] contains a good survey.). The asynchronous and location-oblivious nature of the publish/subscribe models is suitable for content-oriented networking. That is, we can remove the spatial and temporal restrictions of the current Internet [11], [10]. There are two primitives in the publish/subscribe models: *publish* and *subscribe*. Other content-oriented networking proposals (e.g. P2P applications) use the same primitives. For example, DONA uses *REGISTER* and *FIND* primitives for name resolution, which are equivalent to *publish* and *subscribe* primitives, respectively.

In substantiating these two primitives, there are two major alternatives in choosing the underlying infrastructure: a tree and a DHT. In DONA, resolution handlers (RHs)[7] form a tree for the name resolution structure. Even though the tree topology can resemble the relationship between autonomous systems (e.g. provider/customer/peer), it may have problems as follows. The root RH should maintain the location information for every content in the system, but the names (or CIDs) in general cannot be aggregated due to flat CID namespace, i.e. the *routing scalability* problem. Moreover, if we ignore peer links, every link or node in the overlay tree is the single point

[3]Many studies (e.g. [16], [1], [7]) have used their own terminology to represent this paradigm such as *data-oriented*, *content-based*, and *content-centric*. In this paper, these terms are used interchangeably, but we mostly use a term *content-oriented*.

[4]A *CON node* is shortly called a *node* in this paper.

[5]A CON node may or may not have its caching server. In the evaluation section, we assume that every CON node is associated with its own caching server.

[6]Considering incremental deployment, CON nodes should have overlay connections among one another on top of the current IP-based network.

[7]Resolution handlers in DONA correspond to CON nodes in the paper.

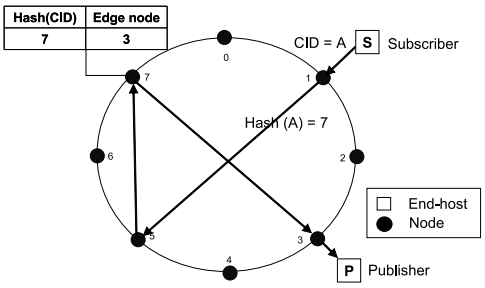

Fig. 2. Resolution architecture based on Chord whose namespace size is 8

of failure. If an RH fails (e.g. due to system failure or DoS attack), the tree is partitioned, which implies the *robustness* problem.

Another alternative that can be used in the name resolution structure is a flat distributed hash table (DHT). First of all, the DHT-based approach lessens the routing scalability problem because of its flatness. For example, assuming Chord [24] is the underlying publish/subscribe model, every CON node needs to know only a small number of nodes (which is the size of the finger table [8]). As for the robustness problem, unlike the tree, each node in the DHT has as many links as the finger table size [19].

The major difference between tree-based and DHT-based resolution infrastructures is how to forward a publish/subscribe message. When an end host publishes a content file in a tree, the publish message is propagated from the host's associated node (or edge node) to the root node. During this propagation, each node creates a routing table entry that maps a CID to its next-hop node towards the edge node. When an end host subscribes to a content file, the subscribe message will be forwarded from its associated edge node to the root node if no nodes along the path has the corresponding entry. The root who has routing information of all contents will forward the subscribe message to the edge node of the publisher. On the other hand, each node in a DHT node forwards the publish or subscribe message to the node whose position in the namespace is the closest predecessor of the hashed value of the CID, which is iterated until the edge node is located. Figure 2 illustrates how a subscribe message for content whose CID is A is processed. Here, S is the subscriber that wishes to retrieve the content file of CID A, P is the publisher, and node 7 holds the location of the edge node (which is node 3) of P.

B. In-Network Caching

By default, we assume that caching is enabled in each CON node, even though non-caching is a possible option. With caching, we can improve the performance of the content delivery by placing the contents closer to subscribers, which comes from the same rationale behind CDNs (e.g. [18], [5]). The usefulness of caching is proven by the success of CDNs in the internet industry. As mentioned in [19], [15], caching is the key factor on the performance of resolution system. Hence, we employ in-network caching as a basic component of content-oriented networking[9].

In our evaluation, each caching server maintains its cache based on the Least Recently/Frequently Used (LRFU) policy [17], which combines the Least Recently Used policy with the Least Frequently Used. The caching mechanism fills the cache starting from the contents with the highest popularity values:

$$F_i = \frac{N_i}{2^{\frac{T_i}{T_{interval}}}} \quad (1)$$

where F_i is the popularity value for content file i, N_i is the number of accesses for content i, T_i is the elapsed time since content file i is cached, and $T_{interval}$ halves the popularity value in order to reflect the pass of time. Everytime a caching server is requested from a CON node to cache a content file and the storage is exceeded, it replaces the contents with the lowest popularity value(s). Equation (1) shows the popularity estimate function, which is a simplified version of LRFU [17]. Sometimes, when caching requests from the node becomes frequent, recently stored content files may not have enough time to increase its access count. To prevent this, we introduce a guard time; a content file i whose T_i value is less than T_{guard} will not be replaced.

C. Delivery Modes

How to deliver contents from a publisher to a subscriber also influences on the overall performance of the content-oriented networking. There are three deliver modes: *end-to-end*, *first-and-last*, and *hop-by-hop*. End-to-end mode refers to the direct delivery from a publisher to a subscriber. In end-to-end mode, a content file is delivered faster than other modes since no other nodes intervene or cache the content file, if not considering the cache effect. In hop-by-hop mode, a content file is relayed by all the CON nodes that have participated in the name resolution process. Note that the content file is passed in the opposite direction along the path over which the subscribe message travels. Therefore, it takes the longest time to deliver the content file; however, the file can be cached in some intermediate nodes (more precisely, the associated caching servers) along the path. The first-and-last mode compromises these two approaches by making only the edge nodes of the publisher and the subscriber relay the contents.

III. Evaluation

We first describe how we configure simulation experiments. Then we compare the performance of two network infrastructures (i.e., the tree and the DHT). We also investigate the impact of caching on the overall performance. There are three

[8]In Chord [24], the authors present that the size of finger table is $log_2 N$, where N is the namespace size in the system.

[9]Obviously, when an autonomous system has multiple caching servers, we can think of coordinated control of caching among them instead of local caching policy. However, in this paper, we assume that each caching server has its own caching policy.

TABLE I
SYSTEM PARAMETERS OF EXPERIMENTS

Common settings	
Number of IP routers	330
Number of nodes	100
Number of published contents	4000
Subscription arrival rate (1/s)	0.5
Cache size of a node	5Gbytes
$T_{interval}$ of cache	3600s
T_{guard} of cache	1800s
Tree-specific settings	
Number of children of a node	4
DHT-specific settings	
Size of namespace	4096
Size of DHT finger table	12

TABLE II
CONTENT TYPES AND SUBTYPES IN PUBLICATION

Content Type	Sub-type	Size (MB)	Portion (%)
VIDEO	Movie	690	3.80%
	TV	92	9.31%
	Animation	71	0.27%
	Porno	72	11.63%
AUDIO	Music	59	23.96%
	Audio book	71	1.04%
SOFTWARE	Application	196	6.55%
	Game	668	6.48%
	Unknown	56	11.97%
WEB	HTTP	3.29	25.00%

modes in content delivery from a publisher to a subscriber. Last but not the least, we explore the performance differences among three delivery modes.

A. Experiment setup

We use ns-2 to evaluate two networking alternatives: a hierarchical tree and a flat DHT. For the purposes of fair comparison, we use the same seed for pseudo random number generation. That is, both the tree and the DHT are tested with the same dataset; i.e., the same CON node subscribes to the same content at the same instance. The end-hosts, which actually publishes or subscribes content files, are assumed to be collocated with CON nodes for simplification. In other words, we assume that the delay between an end-host and its collocated CON node is negligible. There are two options in simulation experiments: caching and non-caching. Caching is enabled by default; we assume that each CON node is collocated with its caching server. Also, there are a few options in content delivery modes as mentioned earlier. When a located file is delivered to the subscriber, we take the hop-by-hop delivery mode by default. Thus, unless otherwise stated, every scenario is tested with caching and in hop-by-hop mode.

In every simulation run for the tree or the DHT, there is an initialization phase. In the initialization phase, each CON node sets up connections with one another. For instance, a parent should have links to its child nodes in the tree, and a node in the DHT should first have links to its successor and predecessor and then populate its finger table entries. After the overlay network setup, we register or publish all the contents to the network. Performance metrics are measured only after the initialization phase; the initialization phase is over at simulation time 36,000 seconds. The overlay network is set up on top of a physical network topology, which emulates the real network topology.

We use the TCP protocol in transmitting content files between CON nodes. Most of TCP parameters in ns-2 are left unchanged, except that the maximum congestion window size is set to 3 MBytes. This may be deemed impractical at this moment. However, considering the ever increasing speed of wired networks, we believe this is a feasible value in the near future. The default system parameters used to our experiments are given in Table I. By default, the results in this section are obtained after 500,000 seconds in simulation time.

1) Traffic Generation: In order to generate the realistic traffic, we first study the recent measurement reports on the real internet traffic. There are four issues in simulating diverse internet contents: types of contents, file size per content type, number of files per content type, and popularity distribution of files per content type. We generate the traffic according to the real traffic measurements in Ipoque [23] with some simplifications. Note that we use slightly different tactics between publishing contents and subscribing contents.

In contents publication, we classify the contents into four types, and each type is subclassified into subtypes as shown in Table II, which is based on Ipoque reports in 2006 and 2007[10]. We consider these four types of contents: video, web, audio, and software. Except for the web traffic, we use the fixed file size for four types and their subtypes from the Ipoque reports. As for the web contents, we analyze a campus network traffic and obtain the size distribution of web content files, which is used in our simulation. We publish a sufficient number of contents for each type, 1000 contents, to prevent the simulation from generating biased results. The portion field in Table II is the ratio of the number of files of a particular content subtype to the whole number of contents.

In contents subscription, we again consider the four content types from Ipoque reports; the volume ratio among video, audio, software, and web contents are 68%, 9%, 9%, and 14%, respectively. As the traffic volume of each content type is approximated by multiplying the number of subscription requests and the average file size, we can calculate the ratio of the number of subscription requests among the four content types. In each content type, the popularity of a particular content file is determined by the widely accepted the Zipf distribution [26]. The parameter of the Zipf distribution is set to 1.0 considering [21].

2) Network Topology: Heckman et al. proposed a method to emulate the existing network topologies of DFN and AT&T by

[10]The reports are available at http://www.ipoque.com/.

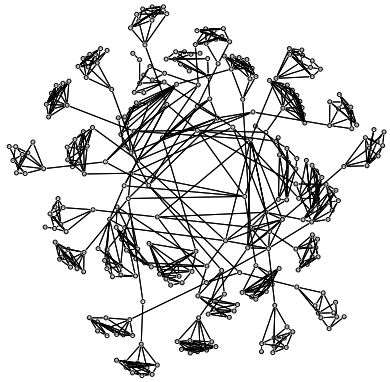

Fig. 3. DFN-like topology with 330 routers

the GT-ITM [2] topology generator [14]. Each generated topology consists of transit networks and stub networks mostly. Based on their study, we generate the physical transit-stub topology that emulates the real network, DFN[11]. Each link has 10Gbps bandwidth capacity. Our physical network topology, which consists of 330 IP routers, is depicted in Figure 3. Among those 330 routers, we randomly select 100 CON nodes from the stub network routers to form the content overlay network in both a tree and a DHT. The reason we exclude the routers in transit networks in selecting CON nodes is that a transit network router typically should forward much more data than a stub network one. Thus, we assume stub network routers are more eligible to be CON nodes, which should perform name resolution and caching as well.

Once the overlay nodes are chosen, the DHT is formed from the overlay nodes by the construction mechanism of Chord [24]. In the case of the tree, we first select the root node and then its child nodes randomly from candidate nodes and so on in a top-down manner. However, in tree formation, we have to consider the maximum number of children of each node. To fairly compare the tree with the DHT, we make the worst-case overlay hop count of the tree to be equivalent to that of DHT. The worst-case overlay hop count of the DHT is 8 due to 4000 content files and 100 CON nodes. That is, the whole namespace is 2^{12} to contain 4000 contents and each node covers $2^{12}/100$ namespace on the average. So, a node needs to travel 7 overlay hops to go to the node that keeps the location of the edge node of the target contents. We add one overlay hop to go to the edge node that actually holds the location of the publisher on the DHT. Thus, the maximum depth of the tree should be four when we consider a case in which a leaf node in the left subtree of the root subscribes to a file in another leaf node in the right subtree of the root. To make a balanced tree, the maximum child nodes of each node in a tree is to be four. Note that a destination node of a subscribe message in the tree is the edge node that holds the location of the publisher; we do not need to add one overlay hop.

[11]We evaluated the performance of content oriented networking alternatives with both of DFN and AT&T topologies, but there was no notable difference. Hence we plot only DFN cases.

B. Network infrastructures

In this section we evaluate the performance of two networking infrastructures in terms of transfer latency, routing scalability, and robustness.

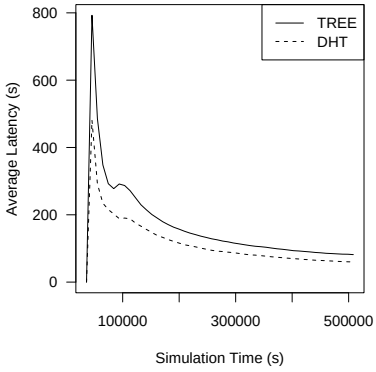

Fig. 4. Transfer latency of the tree and the DHT

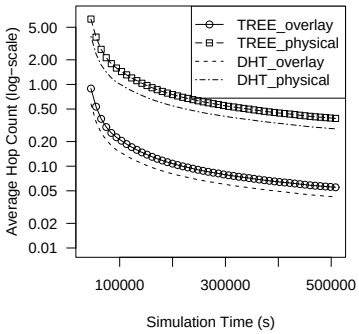

Fig. 5. Hop counts of the tree and the DHT

1) Transfer latency: Figure 4 plots the average latency to transfer contents in the two infrastructures as simulation time goes by. Recall that the evaluation of performance metrics starts at simulation time 36,000 seconds. The latency peaks around simulation time 45,641 seconds and then gradually decreases since in-network caching becomes more and more effective in both infrastructures. Surprisingly, the latency of the tree is substantially higher than that of the DHT. Note that, however, each parent-child link in the tree topology is randomly selected among 100 nodes in the evaluation. We also plot the average hop counts of both overlay and physical paths between the data holder and the subscriber in the tree and the DHT as shown in Figure 5. For both physical and overlay paths, the average hop count in the DHT is shorter than that of the tree. In general, the physical path is proportional to overlay path since every overlay link is randomly selected between two CON nodes. Also, when we analyze the average hop count between an arbitrary pair of nodes in both infrastructures, the tree suffers from longer (overlay) paths since it has more

nodes as the depth (or the distance from the root) increases. That is, for each subscription, a data holder is more likely to be located at lower depth. As time goes on, the latency difference diminishes due to in-network caching. Note that the average hop counts of the overlay paths goes below one since subscribers may be able to fetch the cached contents from their associated edge nodes.

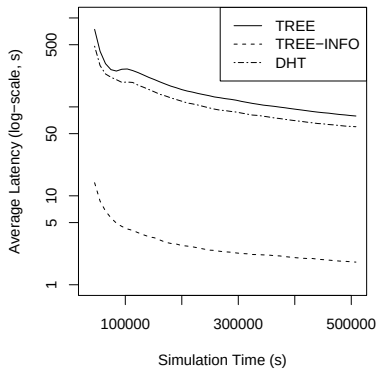

Fig. 6. Transfer latency based on different tree constructions

From Figures 4 and 5, we reach the conclusion that the DHT infrastructure is better than the tree. However, if the tree topology is constructed with the information about the underlying physical network (i.e., a parent node chooses a child node from nearby candidate nodes), the tree outperforms the DHT. Figure 6 shows the transfer latency of the tree with physical network information (labeled "TREE-INFO") is significantly lower than those of the DHT and the tree constructed in a random fashion. We believe a DHT with the physical network information will exhibit better performance, too. Throughout this paper, we evaluate the topology constructed randomly (i.e., both the tree and the DHT) to remove the optimization effect of the information about the physical network.

2) Routing scalability: To investigate the routing scalability of the two structures, we analyze the size of each node's routing table. In the tree, the root should have as many table entries as the number of content files. Thus, its routing table size is the same as the number of contents. When we calculate the average table size of all the tree nodes (i.e. 100 nodes comprise a balanced tree with depth 4), it is proportional to the number of contents. In the DHT, the finger table is the routing table, and its size is determined by the size of namespace, which is assumed to be the number of contents here. It is well known that the finger table increases logarithmically as the namespace increases. Figure 7 shows the results of routing table size of the two infrastructures. Note that the DHT infrastructure scales well with the number of contents.

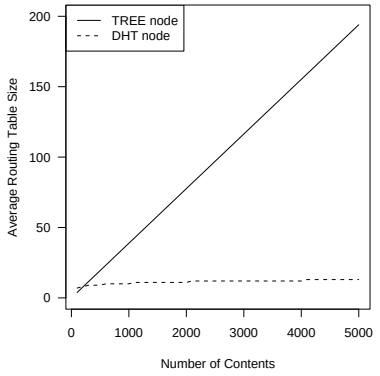

Fig. 7. The average routing table of a tree node increases linearly as the number of contents increases. The routing table (or finger table) size of every DHT node is identical and increases logarithmically.

3) Robustness: In this section, we investigate how much resilient each infrastructure is despite node failures. We assume that CON nodes fail randomly. A crashed node becomes available again after a fixed interval, which is set to one hour in our simulation experiments. One hour downtime simulates the failure of server systems like web servers and application servers as collected in [20]. In simulating the node failures, r is set to a value such that each node is going to fail once during a simulation run on average. Each run is simulated for 300,000 seconds in evaluating robustness. We vary the failure rate from $0.1r$ to $10r$, to compare the resilience of the tree with that of the DHT. When a tree node fails, the tree is partitioned; no subscribe message can go across the failed node. Likewise, when a finger table entry in the DHT relays a subscribe message to a failed node, the corresponding subscription is deemed as a failure. Furthermore, no Chord stabilization is performed in the DHT in the experiments for fair comparison.

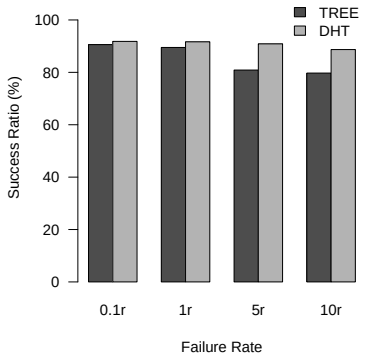

Fig. 8. Robustness of the two infrastructures with caching

Figure 8 shows the success ratio of the subscription requests in the two infrastructures with caching as the node failure rate increases. The DHT's performance gain over the tree is widened as the failure rate increases. Each DHT node has 12 finger table entries (or links to other nodes) and hence it is less vulnerable to node failures. On the contrary, as every tree link is the single point of failure, the tree is much more susceptible to node crashes. Note that the overall success rate is almost above 80% due to caching.

Figure 9 plots the success ratio of the two infrastructures without caching as the node failure rate increases. As there

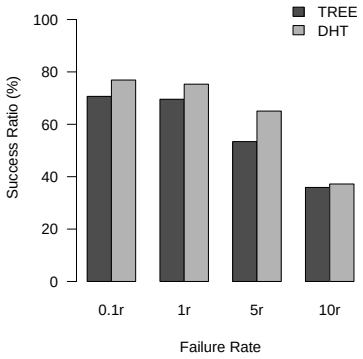

Fig. 9. Robustness of the two infrastructures without caching

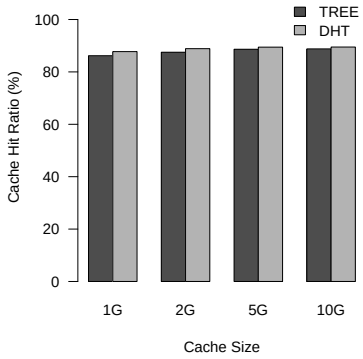

Fig. 11. Cache hit ratio of caching under various storage size

is no caching, the overall success ratio goes even below 40% with the high failure rate. On the average, a subscribe message will go much farther to reach the publisher in this case since intermediate nodes do not cache contents at all. Hence, a subscription request is more likely to fail.

C. In-network caching

In this section, we investigate the cache effect in two scenarios: (i) cache size variation, and (ii) the traffic load (or the subscription request arrival rate). In addition to the transfer latency, we use two more performance metrics: cache utility and cache hit ratio. The cache utility represents how much of the cache storage is occupied with the cached contents; for instance, if a caching server whose storage is 5Gbytes stores 4Gbytes contents at a given instance, the cache utility of the node is 80%. The hit ratio is the ratio of the subscribe messages which incur cache hits to all the subscribe messages generated during an interval. Note that a cache hit happens when any intermediate node in the tree or the DHT has the contents to be requested.

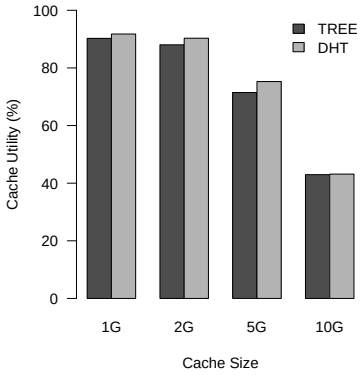

Fig. 10. Cache utility of caching under various storage size

The cache utility and the hit ratio of the two infrastructures are shown in Figures 10 and 11, respectively. Due to the Zipf distribution, the popular contents are more frequently retrieved and cached. Thus, as the cache size increases, it takes more time to "fill" the cache storage with the contents. The cache hit ratio is turned out to be almost independent of both the cache size and the network infrastructure.

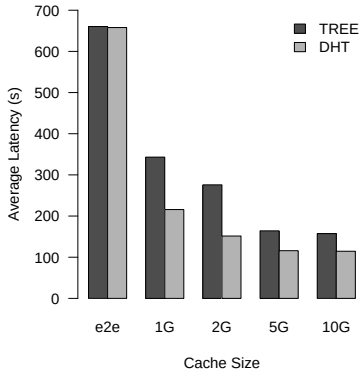

Fig. 12. Delivery latency of caching under various storage size

Obviously, as the cache size increases, the transfer latency becomes reduced as shown in Figure 12. Note that the latency without caching (labeled as "e2e") is measured in end-to-end delivery mode since only end-to-end mode is meaningful when caching is disabled; others are measured in hop-by-hop mode. Caching is should be enabled for performance; the latency without caching is more than twice as high as the one with caching. Also, when the cache storage exceeds 5Gbytes, the performance gain is marginal observing the latency of the 10Gbytes case. Actually the total volume of all the published contents in the network is around 480Gbytes. So, the cache storage in each node is recommended to be around 1% of the total volume to balance between the cache storage cost and the transfer latency.

We vary the arrival rate of subscription requests in order to investigate the impact of the traffic load. Figure 13 shows that the average transfer latency decreases as the traffic load increases. The reason is two-fold: (i) the content overlay network is not overloaded in the tested loads, and (ii) as the traffic load increases, the cache utility increases faster, so that the overall latency is decreased. Figure 14 verifies the second point.

We plot the average latency of four major subtypes (which

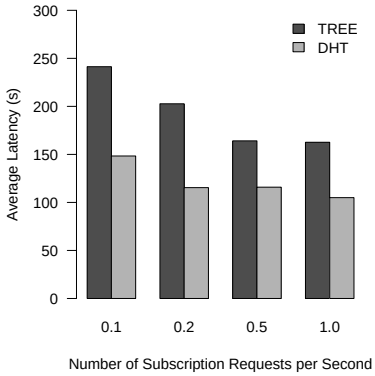

Fig. 13. Average latency of the two infrastructures with caching as traffic load increases.

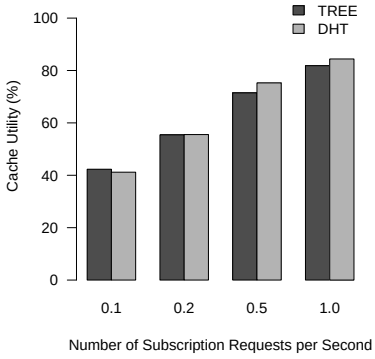

Fig. 14. Cache utility of the two infrastructures with caching as traffic load increases.

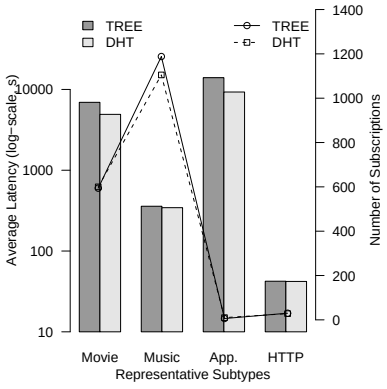

Fig. 15. Average latency of contents is shown; only major subtypes are plotted.

are movie, music, application, and HTTP from Table II) as a bar graph in Figure 15. Note that y axis is drawn in log scale. The latency differences of large contents (movie and application subtypes) between the two infrastructures are wider than those of small contents. Note that the average latency of application contents is higher than that of movie contents despite its smaller size. As the popularity of application subtype contents is much lower than others, its cache effect is trivial. The line graph represents the number of subscribe messages for each subtype. This reaffirms the importance of in-network caching in content-oriented networking infrastructures.

D. Delivery modes

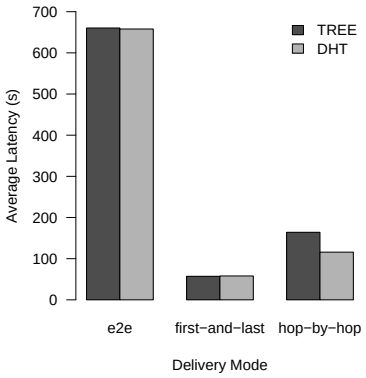

Fig. 16. Average latency in three delivery modes

We compare the performance of three delivery modes: end-to-end, first-and-last, and hop-by-hop. Figure 16 shows the average latency to deliver contents in three modes. Note that the the average latency of the two infrastructures is the same in both end-to-end and first-and-last modes. The reason is that content files in end-to-end and first-and-last modes are delivered over the same physical path. Obviously, the average latency of end-to-end mode is the highest among three modes due to no caching. The average latency of hop-by-hop mode is higher than that of first-and-last mode in both infrastructures since contents will be relayed faster in first-and-last mode.

IV. Related Work

Solutions that have the flavor of content-oriented networking but comply with the current Internet have been or used to be popular. For example, web caching technologies used to provide transparent web services to the end users by processing the requests by looking up local copies [4]. More recently, CDNs [18] manipulate the DNS system to forward the content requests to a close server that has the web contents. They seem to be similar to content-oriented networking in the sense that they improves the system performance by placing the contents closer to end users. However, while the CDN technologies are host-oriented and hence cannot decouple the location of the contents and the contents itself.

One of the recently prevalent applications in the Internet is P2P applications such as BitTorrent [8]. P2P applications provide the content-oriented service; we can retrieve the contents from P2P without any information of hosts that hold the contents. Unfortunately, however, since the P2P applications do not substantiate content-oriented networking at the architectural level. They provide no cross-application abstraction for the content-oriented networking; each P2P application should have its own identifier space, its own service primitives, and its own networking structure, which cannot be shared by other P2P applications.

Carzaniga et al. propose a routing scheme for content-based networking (CBCB) [7]. With adoption of the semantic-aware naming, the CBCB routing mechanism determines the destination of a message not by host-oriented addresses, but by *predicates*. The nodes in CBCB use *predicates*, each of which has the form of an attribute-value pair to represent contents to subscribe. By comparison, DONA [16] proposes a flat (i.e., semantic-less) name for contents, which provides persistence and authenticity. Also it is based on a tree topology and hence performs name resolution in a hierarchical manner. Similarly, Jacobson et al. suggest *content-centric networking* [1] with two claims: (i) the security/authentication should be performed only with data itself, and (ii) the data should be delivered by dissemination, not by endpoint-oriented transport.

In the DNS area, many studies have been performed to compare a hierarchical tree structure with a flat DHT structure, to name a few [9], [19], [22]. They concentrate only on the performance of lookup-by-name resolution system, i.e. DNS, which maps the name to the location of service host. However, for the content-oriented network, the route-by-name paradigm is more appropriate than the lookup-by-name [16], [13]. To the best of our knowledge, this paper is the first comprehensive study to compare the hierarchical networking infrastructure with the flat one under route-by-name resolution environments.

V. Conclusions

Recently, the content-oriented usage such as web, P2P, and multimedia applications becomes more and more proliferated in the Internet. However, the host-oriented original Internet design hinders the proliferation of content-oriented networking applications and services, which motivates new architectural trials such as DONA [16] and content-centric networking [1]. However, the operational issues of a new content-oriented networking framework have not been thoroughly investigated in the literature. We focus on how to substantiate the building blocks of the new networking framework: (i) how to locate contents, (ii) how to deliver contents, and (iii) how to cache contents. There are two major alternatives in choosing the networking infrastructure: a hierarchical tree and a flat distributed hash table (DHT). We carry out comprehensive simulation experiments to compare these alternatives. The DHT achieves the lower latency and better resilience than the tree when the topologies of the infrastructures are randomly built. Caching is crucial in enhancing the performance of both infrastructures. We will investigate how the information about the underlying physical network can optimize the infrastructure construction. Also, the coordination among caching servers will be an interesting issue to reduce the transfer latency.

References

[1] Content centric networking research homepage. http://www.parc.com/research/projects/networking/contentcentric/.

[2] Gt-itm. georgia tech internetwork topology models. http://www.cc.gatech.edu/projects/gtitm/.

[3] Isp-planet - market research - ellacoya's data. http://www.isp-planet.com/research/2007/ellacoya_data.html.

[4] G. Barish and K. Obraczke. World wide web caching: trends and techniques. *Communications Magazine*, 38(5):178–184, May 2000.

[5] M. Beck and T. Moore. The internet2 distributed storage infrastructure project: an architecture for internet content channels. In *Computer Networking and ISDN Systems*, pages 30–22, 1998.

[6] M. S. Blumenthal and D. D. Clark. Rethinking the design of the internet: the end-to-end arguments vs. the brave new world. *ACM Trans. Interet Technol.*, 1(1):70–109, 2001.

[7] A. Carzaniga, M. J. Rutherford, and A. L. Wolf. A routing scheme for content-based networking. In *INFOCOM'04*, Hong Kong, China, Mar. 2004. IEEE.

[8] B. Cohen. Incentives build robustness in bittorrent. In *1st Workshop on Economics of Peer-to-Peer Systems*, 2003.

[9] R. Cox, A. Muthitacharoen, and R. T. Morris. Serving dns using a peer-to-peer lookup service. In *IPTPS*, 2002.

[10] M. Demmer, K. Fall, T. Koponen, and S. Shenker. Towards a modern communications api. In *HotNets VI*. ACM, 2007.

[11] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35:114–131, 2003.

[12] B. Ford, J. Strauss, C. Lesniewski-laas, S. Rhea, F. Kaashoek, and R. Morris. Persistent personal names for globally connected mobile devices. In *OSDI '06*, pages 233–248, 2006.

[13] M. Gritter and D. R. Cheriton. An architecture for content routing support in the internet. In *the 3rd Usenix Symposium on Internet Technologies and Systems (USITS)*, pages 37–48, 2001.

[14] O. Heckmann, M. Piringer, J. Schmitt, and R. Steinmetz. On realistic network topologies for simulation. *the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, pages 28–32, 2003.

[15] L. Iannone and O. Bonaventure. On the cost of caching locator/id mappings. In *CoNEXT '07*, pages 1–12. ACM, 2007.

[16] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. In *SIGCOMM '07*, pages 181–192, 2007.

[17] D. Lee, J. Choi, J. Kim, S. Noh, S. Min, Y. Cho, and C. Kim. Lrfu: A spectrum of policies that subsumes the least recently used and least frequently used policies. *IEEE Transactions on Computers*, 50(12):1352–1361, 2001.

[18] G. Pallis and A. Vakali. Insight and perspectives for content delivery networks. *Communications of ACM*, 49(1):101–106, 2006.

[19] V. Pappas, D. Massey, A. Terzis, and L. Zhang. A comparative study of the dns design with dht-based alternatives. *INFOCOM '06*, pages 1–13, April 2006.

[20] S. Pertet and P. Narasimhan. Causes of failure in web applications. Technical Report CMU-PDL-05-109, Parallel Data Laboratory, Carnegie Mellon University, 2005.

[21] V. Ramasubramanian. Beehive: Exploiting power law query distributions for o(1) lookup performance in peer to peer overlays. In *NSDI '04*, 2004.

[22] V. Ramasubramanian and E. G. Sirer. The design and implementation of a next generation name service for the internet. In *SIGCOMM '04*, pages 331–342. ACM, 2004.

[23] H. Schulze and K. Mochalski. ipoque - internet study 2007 data about p2p, voip, skype, file hosters like rapidshare and streaming services like youtube. http://www.ipoque.com/news_&_events/internet_studies/internet_study_200%7/.

[24] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01*, pages 149–160. ACM, 2001.

[25] M. Walfish, H. Balakrishnan, and S. Shenker. Untangling the Web from DNS. In *NSDI '04*, San Francisco, CA, March 2004.

[26] G. K. Zipf. *Selected Studies of the Principle of Relative Frequency in Language*. Harvard University Press, 1932.