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

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

***Abstract*—While the Internet was designed with host-oriented networking applications, recent Internet statistics show that content-oriented traffic has become more and more dominant. Even though content-oriented networking, which tries to resolve this discordance, has received increasing attention, there have been few comprehensive and quantitative studies on how to realize a content-oriented networking architecture. In this paper, we focus on the design alternatives of the content-oriented networking architecture and evaluate their performance: (i) how to locate contents, (ii) how to cache contents, and (iii) how to deliver 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.**

***Index Terms*—Network Architecture and Design, Name Resolution, Route-by-name Paradigm, Contents Caching, Performance Evaluation**

I. Introduction

One of the Internet design principles that hinders the current Internet evolution is the host-oriented communications. Over the past several years, however, the vast majority of Internet usage has become data retrieval and service access [7], [3] (i.e., *data-oriented* usage), where an end user cares more about contents but is oblivious to or less aware of the host (or its location) [3]. The gap between host-oriented Internet architecture and the data oriented usage has limited the data availability, ease-of-use, performance, and scalability of Internet applications.

To name a few, search engines, web caching, content delivery networks (CDNs) [5], and P2P applications seek to support data-oriented usage by manipulating DNS naming and/or name resolution. Recently, lessons from these several application/service-dependent ad-hoc solutions, have motivated "clean-slate" efforts [3], called data-oriented or content-centric networking, which strive to redesign the Internet architecture to accommodate the dominant data-oriented Internet usage. The essence of content-oriented networking lies in decoupling contents from the service host/location at networking level, not at application level. In this way, the content-oriented networking architecture will (i) free application/service developers from re-inventing the same set of mechanisms required to support data-oriented usage, and thus (ii) enable scalable and efficient content delivery.

Koponen *et al.* proposed the Data-Oriented Network Architecture (DONA [3]), in which they suggest the use of persistent and self-certifying content identifiers and the hierarchical tree resolution infrastructure based on the *route-by-name* paradigm for the content-oriented networking. However, the tree resolution infrastructure of DONA has limited routing scalability since the flat content identifier cannot be aggregated following the tree topology in contrast to the hierarchically structured domain names of the DNS. Moreover, the number of content files will be huge and the root node should maintain as many resolution entries as the number of content files in the system.

To resolve the routing scalability problem, we turn our attention to a flat resolution infrastructure using a distributed hash table (DHT). DHT-based resolution will mitigate the routing scalability problem substantially, compared to the tree-based resolution. However, other performance metrics of tree- and DHT-based resolution infrastructures should be investigated. Therefore, in this paper, we investigate which of DHT and tree resolution infrastructures are suitable for the content-oriented networking in comprehensive scenarios. To the best of our knowledge, this paper is the first work that quantitatively evaluates and compares the performance of these two alternatives based on the the route-by-name paradigm of content-oriented networking.

II. Content-oriented Networking

A *content-oriented network* (CON) is a new network architecture designed to operate under the data-oriented paradigm There are three kinds of entities in a CON: (i) *CON nodes*, (ii) *caching servers*, and (iii) end hosts. *CON nodes* form an overlay for content-oriented networking for end hosts. When a content request message arrives at a CON node, it looks up its *caching server*. If cache hits, the CON node will deliver the content file. *End hosts* are connected to CON node(s) and request/deliver the contents.

A. Name Resolution

There are two name resolution mechanisms: *lookup-by-name* and *route-by-name*. "Lookup-by-name" refers to a process to find out an IP address with the name. On the other hand, "route-by-name" means how to locate a corresponding host, which holds the content file, with the name. We now

elaborate on the latter, which is more appropriate in content-oriented networking since it decouples the content and its location. Also, we adopt the publish/subscribe model for content-oriented networking in order to remove the spatial and temporal restrictions of the current Internet.

In substantiating *publish* and *subscribe*, there are two major alternatives: a tree-based and a DHT-based ones. In DONA, resolution handlers (RHs) , which correspond to CON nodes, form an overlay tree of the name resolution structure. The root RH should maintain the mapping information of every content in the system, but the names in general cannot be aggregated due to the flat namespace. Assuming the even distribution of contents, each node maintains the mapping information of at least C/N contents, where C represents the number of contents and N represents the number of nodes. As a parent node should keep all the mapping information of its child nodes, a node whose height is h in a m-ary complete overlay tree should maintain $C/N \cdot \sum_{i=0}^{h} m^i$ routing entries. Moreover, every link or node in the overlay tree is a potential single point of failure or bottleneck.

An alternative to the tree structure is a DHT-based structure. It lessens the routing scalability problem due to a shared hash function: A DHT node has $log_2 C$ routing table entries and C/N resolution table entries. As for the robustness problem, unlike the tree, every node in the DHT makes the same contribution as others (i.e., no single point of failure or bottleneck).

B. In-Network Caching

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. In our evaluation, each caching server uses the Least Recently/Frequently Used (LRFU) replacement policy [4]. The LRFU policy 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 since it is stored, 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. To provide the enough change for the recently cached files, we introduce a guard time T_{guard}.

For the performance of in-network caching, it is important to trade-off between increasing the probability of cache hit and minimizing the content delivery path. To explore this trade-off, we compare three delivery modes: *end-to-end*, *first-and-last*, and *hop-by-hop*. End-to-end mode refers to the direct delivery from a content holder to a subscriber. In hop-by-hop mode, a content file is relayed by all the nodes that have participated in the name resolution. The first-and-last mode compromises these two approaches by making only the edge nodes of the content holder (in case that the holder is an end-host) and the subscriber relay the contents and get a chance to cache.

III. Evaluation

A. Experiment setup

We use ns-2 to evaluate two alternatives: a tree-based and a DHT-based structures. The end-hosts are assumed to be collocated with nodes for simplification. There are two options in simulation experiments: caching and no caching. And caching is enabled by default. Also, when a located file is delivered to the subscriber, we take the first-and-last delivery mode by default.

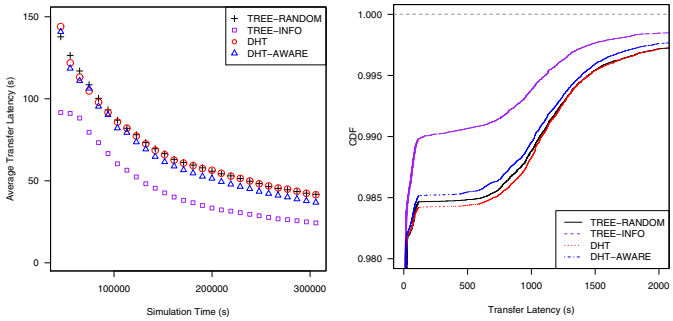
In order to generate the realistic traffic, we first study the recent measurement reports on the real Internet traffic. There are four issues in simulating diverse contents: types of contents, file size per content type, number of files per content type, and popularity distribution of files per content type. We generate the traffic according to the Ipoque's [7] Internet traffic measurement report with some simplifications. Note that we use slightly different tactics between publishing contents and subscribing contents.

Heckman *et al.* proposed how to emulate two existing physical networks by the GT-ITM topology generator [2]. Based on their study, we generate the physical transit-stub topology, which consists of 312 IP routers and whose link has 10Gbps bandwidth capacity, that emulates the real AT&T network. Among those 312 routers, we randomly select 100 CON nodes to form the content overlay network. Once the overlay CON nodes are chosen, there are two ways to construct a tree. If information about the underlying physical network is available, a parent node can choose its children from nearby nodes. Otherwise, the parent randomly chooses its child nodes. The former will be denoted by *TREE-INFO*, and the latter by *TREE-RANDOM*. We use both topologies in evaluation to investigate the effect of how to construct the tree on the transfer latency. Note that we use only TREE-RANDOM in comparing with DHT for other experiments (i.e., TREE represents TREE-RANDOM). For tree construction, we should 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. In constructing a DHT, there are two ways: (i) *DHT* and (ii) *DHT-AWARE*. The DHT is formed from the overlay nodes by the original construction mechanism of Chord [8]. In the DHT-AWARE, a node periodically measures the network delays of candidates for each of its finger table entry and selects the one with lowest delay.

B. Two Infrastructures

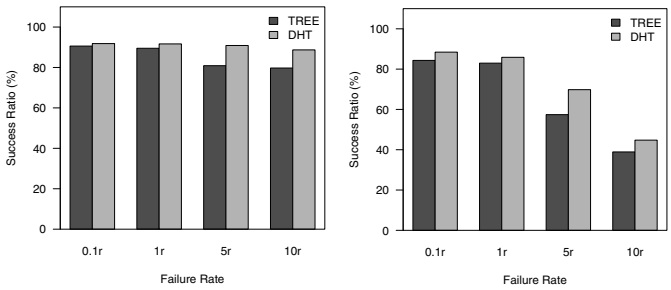
In this section we evaluate the performance of two content-oriented networking infrastructures in terms of transfer latency and robustness.

1) Transfer latency: Figure 1(a) plots the average transfer latency of the two infrastructures as simulation time goes by. Recall that the performance evaluation starts at simulation time 36,000 seconds. The latency peaks around simulation time 46,000 seconds and then gradually decreases as in-network caching becomes more effective in both infrastructures. Figure 1(b) shows the cumulative distribution function (CDF)



(a) Transfer latency (b) CDF of transfer latency

Fig. 1: Delivery performance between Tree and DHT



(a) With caching (b) Without caching

Fig. 2: Robustness of the two infrastructures

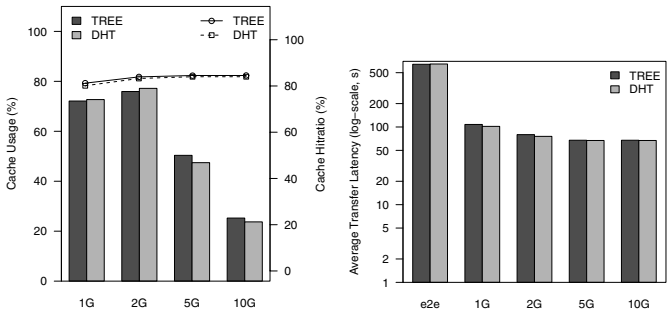
of the transfer latency for all the delivered contents in the four topology constructions. The latency results of the tree topology built without any physical network information are higher than those of the physical topology-aware one (i.e., TREE-RANDOM achieves worse latency than TREE-INFO). The reason is that the data holder is often "physically" closer to the subscriber in the TREE-INFO topology construction since every tree link is set up between two close nodes. In contrast, the performance gain (in terms of latency) of the DHT-AWARE construction over the DHT construction is marginal. In a DHT-AWARE topology in the experiments, even though the next hop node in each finger table entry is set up considering the physical distance, the physical distance between two nodes becomes negligible as the overlay hop count between them in the DHT increases. Thus, the probability that the data holder is "physically" closer to the subscriber in DHT-AWARE construction is trivial. From this, one might conclude that the DHT infrastructure cannot benefit from the network topology-aware construction not so much as the tree infrastructure. However, if we use a DHT that is more essentially constructed with physical distance information (e.g. [1]), we believe the performance gain of the DHT-AWARE topology over the original DHT can be noticeable.

2) Robustness: In this section, we investigate how much resilient each infrastructure is in the face of random node failures. A crashed node becomes available again after a fixed interval (one hour in simulation). 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. 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 messages can go through 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 for fair comparison.

Figure 2 shows the successful delivery ratio of the two infrastructures with/without caching as the node failure rate increases. The DHT's performance gain over the tree is noticeable in comprehensive settings. 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 a potential single point of failure, the tree is more susceptible to node crashes. The performance gain of the DHT is noticeable but not so significant. So, which infrastructure between the tree and the DHT is deployed is not so crucial for resiliency; instead, it is more effective to use other measures (e.g. backup nodes). Comparing Figure 2 (a) and (b), we find that in-network caching substantially increases the system resiliency as the failure rate increases.

C. In-network caching

In this section, we investigate the cache effectiveness in varying the storage size for caching. In addition to the transfer latency, we use two performance metrics: cache usage and cache hit ratio. The cache usage represents how much of the cache storage is used to cache the contents; for instance, if a caching server whose storage is 5Gbytes stores 4Gbytes contents at a given instance, the cache usage of the node is 80%. The hit ratio is the ratio of the subscribe messages which incur cache hits to all the generated subscribe messages during an interval. Note that a cache hit occurs when any intermediate node in the tree or the DHT has the requested contents.



(a) Cache usage and Hit ratio (b) Delivery latency

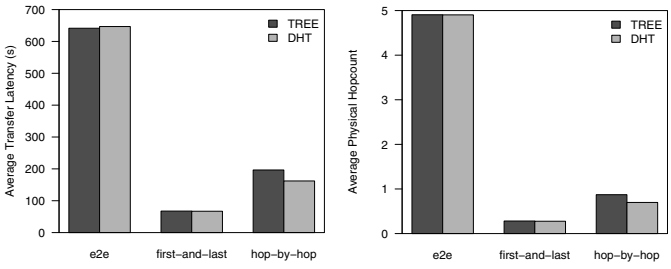
Fig. 3: Performance with varying storage size

The cache usage and hit ratio of two infrastructures are shown in Figures 3(a), a bar graph plots cache usage and the line shows cache hit ratio. Due to the Zipf distribution, the popular contents are more frequently retrieved and cached.

Thus, as the cache size increases, the less portion of the storage is used for contents caching. Note that the cache usage of 1 GBytes case is less than that of 2 GBytes. The reason is that a video file is 690 MBytes. Hence, when a video file is replaced, the cache usage is lowered significantly. The cache hit ratio is turned out to be almost independent of both the cache size and the network infrastructure (tree versus DHT).

Obviously, as the cache size increases, the average transfer latency is reduced as shown in Figure 3(b). Here y axis is in log scale. 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 first-and-last mode. Caching should be enabled for better performance; the latency without caching is at least six times as high as the one with caching. Also, when the cache storage exceeds 5Gbytes, the performance gain becomes marginal. Actually the total volume of all the published contents in the network is around 480Gbytes. So, we can improve the system performance remarkably if each node keeps cache whose size is no more than 1% of the total volume.

D. Delivery modes



(a) Average latency (b) Average physical hop counts

Fig. 4: Performance of three delivery modes

We compare the performance of three delivery modes: end-to-end, first-and-last, and hop-by-hop. Figure 4(a) shows the average latency to deliver contents in three modes. Note that the average latency of the two infrastructures (DHT and TREE) is almost same in both end-to-end and first-and-last modes. The reason is that content files in end-to-end mode are delivered over the same physical path, though the resolution latency is different between the tree and the DHT. In first-and-last mode, the physical paths are the same if no cache hit happens at intermediate nodes.

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 are cached at much more intermediate nodes in hop-by-hop mode than in first-and-last mode (see Figure 4(b)). This gives us a lesson that caching contents at too many places is not useful.

IV. Conclusions

To resolve the discordance between the host-oriented design of Internet and its content-oriented usage, the content-oriented networking is introduced in the literature, featuring the persistent and self-certifying content identifiers and route-by-name resolution infrastructure. However, the operational issues of a new content-oriented networking architecture have not been thoroughly investigated. In this paper we mainly focused on three issues of content-oriented networking: (i) how to locate contents, (ii) how to deliver, and (iii) how to cache. DONA [3] suggests a tree-based resolution infrastructure, which is not scalable with the number of contents. Thus, we compared it with a DHT structure, which is known to be scalable. We analyzed the pros and cons of the tree and DHT structures quantitatively in terms of the routing scalability, robustness, and transfer latency. If the network will be deployed with information about underlying physical network information, the tree structure will be preferred because of its superior delivery efficiency. Otherwise, the DHT structure is better because of its superior scalability and better resiliency. In addition, we found that caching is crucial in enhancing the performance of both structures from experiments; the delivery latency and the resiliency of system are improved due to caching.

Acknowledgment

This work was supported by the IT R&D program of MKE/IITA [2007-F-038-03, Fundamental Technologies for the Future Internet]. This research was also supported by the Ministry of Knowledge Economy, Korea, under the Information Technology Research Center support program supervised by the Institute of Information Technology Advancement (grant number IITA-2009-C1090-0902-0006). The ICT at Seoul National University provides research facilities for this work.

References

[1] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Exploiting network proximity in distributed hash tables. In *International Workshop on Future Directions in Distributed Computing (FuDiCo)*, pages 52–55, 2002.

[2] 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.

[3] 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.

[4] 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.

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

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

[7] 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_2007/.

[8] 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.