Is Information-Centric Multi-Tree Routing Feasible?

Antonio Carzaniga, Koorosh Khazaei, Michele Papalini
University of Lugano
Lugano, Switzerland

Alexander L. Wolf
Imperial College London
London, United Kingdom

ABSTRACT

We have argued that an information-centric network should natively support publish/subscribe event notification in addition to on-demand content delivery. We have also argued that both primitives could use the same forwarding information base and, furthermore, that both primitives can easily support addresses that are more expressive than simple hierarchical names. In this paper we present a concrete routing scheme that realizes this: "push" as well as "pull" communication; anycast as well as multicast; and descriptor-based (as opposed to name-based) addressing. The scheme is founded on multiple tree covers that can be arranged and composed hierarchically following the structure of network domains. On each tree, the scheme combines addresses so as to reduce forwarding state. We demonstrate the feasibility and scalability of the scheme through simulations on Internet-scale workloads in realistic network settings.

Categories and Subject Descriptors

C.2.1 [**Computer Systems Organization**]: Computer-Communication Networks—*Network Architecture and Design*

General Terms

Design

Keywords

Publish/subscribe; routing; content-based, content-centric, named-data, and information-centric networking

1. INTRODUCTION

Information-centric networking is based on the general notion that the network should offer communication primitives using addresses on information rather than on (host) locations. Within this conceptual framework, the primitives may vary significantly in the way that they identify and provide access to information.

ICN'13, August 12, 2013, Hong Kong, China.

One way to obtain information is for the consumer to "pull" data: a consumer requests some information; the network forwards the request to one or more producers that are willing to provide that information; those producers respond with a data packet that satisfies the request; and the network forwards the data packet back to the consumer. We refer to this as *on-demand content delivery*. Another way is for the producer to "push" data: a producer publishes a data packet that the network forwards to the consumers willing to receive that information. We refer to this as *publish/subscribe event notification*.

Both kinds of primitives must somehow identify information. One way is to *name the data*, which can be done with either flat or hierarchically structured *names*, with exact or prefix matching semantics, respectively. An alternative way to identify information is to *describe its content*, which would be done using some form of *descriptors* rather than names. For example, one could use simple character string "tags". The data would be associated with a set of tags and could be referred to by any subset of those tags. In the remainder of this paper we assume the use of character string tags, although other forms of descriptors could be considered [2].

In a previous paper we argued that a truly information-centric network should natively support on-demand content delivery as well as publish/subscribe event notification [1]. The former corresponds to the notion of "content-centric networking" or "named-data networking" (NDN) [7], while the latter corresponds to the notion of "content-based networking" [3, 4, 5]. Furthermore, we argued that both primitives should use the more expressive descriptor form of addressing, and that both could be beneficially implemented together through the same forwarding information base (FIB).

In this paper we first present a routing scheme that substantiates our earlier arguments. We then analyze its scalability by considering plausible network-wide workloads. The scheme supports content delivery and event notification, and can be used with either or both name-based and descriptor-based addresses. In fact, the scheme is virtually agnostic with respect to the addressing scheme and, yet, admits to compact address aggregation. No underlying network service is innately required beyond the basic connectivity of neighbor routers, but advantage could be taken of a unicast network service such as IP, if available. Furthermore, no per-packet or per-flow network state is required; this in contrast to the proposed NDN routing scheme [7], which requires a table of pending interests. Lastly, the scheme uses a unified FIB that can be compiled using standard routing protocols and according to various network design objectives.

2. NETWORK MODEL

As we have argued, on-demand content delivery and publish/subscribe event notification have fundamental commonalities that suggest a synergistic implementation in a unified content-based network [1]. We now describe the architecture of such a network.

We start with network addresses: *descriptors*. Descriptors correspond to the names and prefixes of the NDN network model [7]. Thus, they are associated with information of interest. Furthermore, they are used to express *immediate* interests that represent requests in on-demand content delivery, and *continuing* interests that represent subscriptions in publish/subscribe event notification.

More specifically, in on-demand content delivery, descriptors play three roles: (1) a producer registers one or more descriptors that identify the data that the producer is willing and able to provide; (2) a consumer requests a data packet by issuing an interest carrying a descriptor that specifies the requested data packet; and (3) a producer responds to an interest by returning a data packet (i.e., the content) carrying a descriptor that identifies the data. In publish/subscribe event notification, descriptors play two roles: (4) a consumer registers one or more descriptors that specify the data that the consumer wishes to receive and (5) a producer issues a data packet carrying a descriptor that identifies the data. To avoid ambiguities, we refer to these five roles, respectively, as producer *offer*, consumer *request*, producer *data reply*, consumer *subscription*, and producer *notification*; we avoid the term "interest" to avoid confusion between immediate interests (requests) and continuing interests (subscriptions).

The semantics of descriptors determine the matching relations between descriptors in their various roles: how data replies match requests, how offers match requests (and, therefore, how offers describe the data available from a producer), and how notifications match subscriptions.

As discussed so far, descriptors are abstract and generic. Indeed, much of what we propose is conceptually independent of their specific form and semantics. However, we must be specific in order to develop a concrete routing scheme. For this purpose we adopt "tags".

A descriptor consists of a set of string tags, with the matching relations corresponding to the intuitive subset relations between sets of tags. Specifically, a descriptor D in a data reply would match a descriptor R in a request when the reply contains all the tags of the request, that is, when $D \supseteq R$. Consistently, a descriptor R in a request would match a descriptor O in an offer when the request contains all the tags of the offer, and thus $R \supseteq O$. Also consistently, a descriptor N in a notification would match a descriptor S in a subscription when the notification contains all the tags of the subscription, and thus $N \supseteq S$.

Tag-based descriptors illuminate the commonalities of on-demand content delivery and publish/subscribe event notification, suggesting an implementation of both using a unified FIB. For on-demand content delivery, the FIB directs *requests* toward hosts that are willing and able to satisfy them (with corresponding data replies); for publish/subscribe event notification, the FIB directs *notifications* toward hosts that are willing to receive them. So, both forms of communication allow hosts to declare which messages they intend to receive—requests in on-demand content-delivery and notifications in publish/subscribe event notification—and their difference is simply the *source* of the routing information—

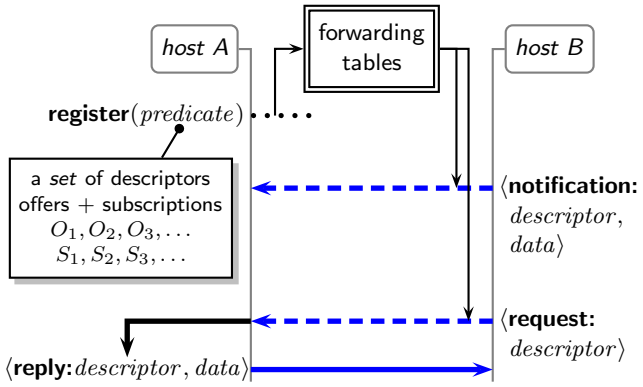

Figure 1: Unified Content-Based Network Layer

producers in on-demand content-delivery and consumers in publish/subscribe event notification.

In terms of data traffic, the difference between the two primitives is a bit more involved. Both requests and event notifications are forwarded along paths toward matching descriptors. However, an interest is expected to generate a corresponding reply, while an event notification is a one-way message. Furthermore, the caching semantics are different. A request that can be satisfied by cached content will not be forwarded downstream toward the original producer node, while an event notification must be forwarded all the way to interested consumers (although the event notifications might be cached for reliability purposes).

In summary, a common content-based layer is configured through the registration of descriptors that define a descriptor-matching *predicate* as a host address, and exploited using three types of messages: one-way notifications ("push"), requests that expect a reply ("pull"), and data replies. This network interface is illustrated in Figure 1.

3. ROUTING SCHEME

We propose a routing scheme based on multiple trees. In essence, the scheme is a simple routing scheme on a tree cover, extended with multiple trees within the same network domain and across network domains. We start by describing the scheme on a single tree.

Consider a network spanned by a tree T. T is identified within each notification and request packet so that each router v can determine the set adj_v^T of its neighbors that are also adjacent to v in T. This can be done by adding an identifier for T in the packet and storing the adjacency set adj_v^T at each router v, or in a completely stateless manner by encoding the whole tree T within a packet header [8].

The FIB of router v associates each neighbor w in adj_v^T with the union $P_{T,w}$ of the predicates advertised by all the hosts reachable through neighbor w on T (including w). An example is shown in Figure 2.

Given a FIB representing $P_{T,w}$ for each adjacent router w in adj_v^T, forwarding proceeds intuitively as follows: Router v forwards a notification with descriptor N received from neighbor u to all neighbors $w \neq u$ in adj_v^T whose associated predicate $P_{T,w}$ matches N. (A predicate P matches a descriptor X if any one of the descriptors in P matches X.) Similarly, router v forwards a request with descriptor

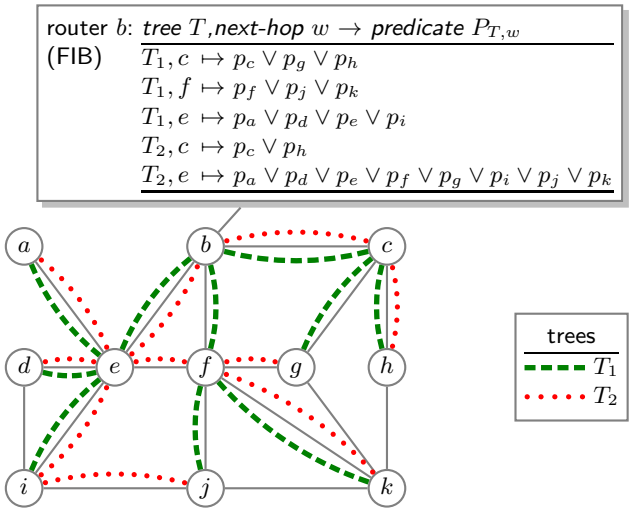

Figure 2: Multi-Tree Routing Scheme

R received from neighbor u to one neighbor $w \neq u$ in adj_v^T whose associated predicate $P_{T,w}$ matches R.

3.1 Using Multiple Trees

Routing on a tree cover has two disadvantages. First, paths might be "stretched", meaning the distance between two nodes on the tree might be longer than on the full graph. For example, in tree T_1 in Figure 2, the distance between nodes i and j is as high as 4 hops, whereas the distance in the full graph is only 1 hop. Second, traffic would flow only on the tree, thereby reducing the overall network throughput.

These problems can be alleviated by using multiple trees, each with their own forwarding state. A notification or request is committed to, and thereafter routed using, one of those trees. Multiple trees may be interchanged by gateway routers in a hierarchical routing scheme. The choice of trees, both in the way they are built and in the way they are assigned by gateway routers, can be used to implement various routing strategies. For example, a domain hosting a popular content producer may build and use one or more shortest-paths trees rooted at that producer so as to reduce latency and spread traffic.

3.2 Memory Complexity and Implementation

A network in which addresses consist of descriptors chosen by applications (as opposed to identifiers chosen by the network provider) raises a question of scalability for the FIB. The central question is to what extent addresses aggregate. Furthermore, there is a question of cost for maintaining FIBs representing multiple trees. We first discuss aggregation of predicates within a single tree, and then discuss aggregation across trees. Lastly, we present an efficient implementation of compact FIBs.

3.2.1 Destination Grouping

Descriptors aggregate in a way that is analogous to the aggregation of IP prefixes [4]: predicates registered for different destination hosts may lead to an aggregated representation when they are associated with the same interface in a FIB. This amounts to representing a logical disjunction of predicates with a more compact representation than the simple list of individual predicates, similar to per-interface aggregation of group addresses in IP multicast [10]. Specifically, a descriptor X subsumes all other descriptors Y that contain X. For example, if two applications register subscriptions $\{networking, conference\}$ and $\{networking\}$, respectively, and if both are reachable through the same interface, then a router may ignore the first descriptor, since any notification matching the first would also match the second. In Section 4 we show experimental evidence that descriptor-based predicates aggregate well in practice.

3.2.2 Tree Grouping

In general, multiple trees require more forwarding state. In fact, there is a trade off between the cost of increased forwarding state and routing complexity, on the one hand, and path stretch and congestion on the other. At one extreme, the case of a single tree incurs the worst stretch and congestion. At the other extreme, one shortest-path tree rooted at each source (so that each notification or request could be routed on their absolute shortest paths) incurs a large amount of forwarding state.

This trade off has been studied extensively from a theoretical perspective, but such studies have had arguably little impact on the practical design of Internet routing, perhaps because even an asymptotically constant stretch is considered unacceptable. Our intuition, confirmed by the experimental analysis presented in Section 4, is that in practice near-optimal paths are possible at a reasonable price in terms of forwarding state.

A first idea is to aggregate *across* trees. Observe that, depending on the choice of trees, and sometimes depending only on the network topology, some trees might partially overlap. If trees overlap, or even if they do not overlap completely but they share the same mapping of destinations to interfaces on some routers, then those trees are *indistinguishable* for the purpose of forwarding at those routers. This means that those routers can collapse the otherwise separate representations of the two trees in their FIBs.

However, as it turns out, computing and maintaining the indistinguishability between trees is complex, also because the relation may be different for each router. And doing so purely on the basis of the network topology, ignoring the association of predicates to trees, may be ineffective. Therefore, our second and more interesting idea is to combine the aggregation of trees with, and transform into, the aggregation of predicates. We achieve this with a simple and yet effective multi-tree FIB transformation that we call *table folding*.

3.2.3 Table Folding

A naive implementation of the FIB would amount to a separate forwarding table for each tree. This is illustrated in Figure 3 (top) using the example network of Figure 2. Notice that such a structure maintains a separate per-source, per-branch mapping of predicates. If, instead, we could somehow fold the tables together by joining the predicates into larger disjunctions, then we can create new opportunities to perform logical simplifications that lead to smaller memory representations.

In our scheme we do this by artificially creating a new tag that we call the *tree tag* and incorporating this tag into per-link (i.e., per-interface) predicates. The result is a folded table exemplified in the bottom of Figure 3. In the folded

FIB of router b (Figure 2)	
tree T, *next-hop* w	$\rightarrow$ *predicate* $P_{T,w}$
T_1, c	$\rightarrow p_c \vee p_g \vee p_h$
T_1, f	$\rightarrow p_f \vee p_j \vee p_k$
T_1, e	$\rightarrow p_a \vee p_d \vee p_e \vee p_i$
T_2, c	$\rightarrow p_c \vee p_h$
T_2, e	$\rightarrow p_a \vee p_d \vee p_e \vee p_f \vee p_g \vee p_i \vee p_j \vee p_k$

$\Downarrow$

FIB of router b (Figure 2)	
next-hop w	$\rightarrow$ *predicate* $P_{*,w}$
c	$\rightarrow p_c \vee p_h \vee (\text{“}T_1\text{”} \wedge p_g)$
f	$\rightarrow (\text{“}T_1\text{”} \wedge p_f) \vee (\text{“}T_1\text{”} \wedge p_j) \vee (\text{“}T_1\text{”} \wedge p_k)$
e	$\rightarrow p_a \vee p_d \vee p_e \vee p_i \vee (\text{“}T_2\text{”} \wedge p_f) \vee (\text{“}T_2\text{”} \wedge p_g) \vee (\text{“}T_2\text{”} \wedge p_j) \vee (\text{“}T_2\text{”} \wedge p_k)$

Figure 3: Folding Tree Selection into Predicate Matching

table we use the abbreviated notation ($\text{“}T_i\text{”} \wedge p_x$) to refer to a predicate consisting of all the descriptors in p_x, each with an added tree tag “T_i”. Correspondingly, the same special tree tag “T_i” is added to all notifications and requests forwarded along tree T_i. Notice that the folded table must associate a predicate ($\text{“}T_i\text{”} \wedge p_x$) to an interface w for each tree T_i in which predicate p_x is associated with neighbor w. This is the case, for example, for predicate p_a and neighbor e in Figure 3. However, since p_a is associated with neighbor e on *all* trees (T_1 and T_2), then the folded FIB can use the simple predicate p_a.

Notice how tree aggregation translates into a simple matching problem, and how the translation allows for logical simplification and compression of predicates. Specifically, notice that adding tree tags to all predicates amounts to exploiting the indistinguishability of trees implicitly.

In practice, for the experimental evaluation presented in Section 4, we use an implementation of forwarding tables that performs predicate aggregation by removing redundant descriptors. Also, in order to obtain a compact representation, we use Bloom filters to represent tag sets.

4. EVALUATION

We now present the results of an experimental analysis of the scalability of the proposed multi-tree routing scheme. Beyond this, we also attempt to answer the more fundamental question of the scalability of routing in information-centric networks. Specifically, the two main questions we pose are: (**Q1**) to what extent is it possible to use trees to route traffic over the Internet and (**Q2**) to what extent do user-defined descriptor-based addresses aggregate at the global scale. The first question leads to a conceptually simple analysis, since it examines some topological properties of the current Internet. The second question is conceptually more complex, primarily because it involves many more unknowns, since it requires the analysis of plausible *future* usage patterns and, in particular, the analysis of the ways in which applications would describe and access information.

4.1 Internet Topology and Trees

As we argue in Section 3.2.2, trees pose a trade off between traffic overhead and the cost of processing and storing multiple trees. Thus, on the one hand, we want to characterize the traffic overhead as a function of the number of trees and, on the other, we want to characterize the memory costs and the ability to aggregate multiple trees.

We conduct our analysis on the Internet AS-level topology[1] consisting of a graph of 42113 nodes and 118040 edges. We then simulate our scheme by constructing sets of k tree covers as follows: We choose k nodes at random by selecting Tier-1, Large ISP, Small ISP, and Stub ASes with probability 40%, 30%, 20%, and 10%, respectively, where Large ISP, Small ISP and Stub are ASes with a customer tree of 50 or more, between 5 and 50, and less than 5, respectively. We then use k shortest-paths trees, each rooted at one of the k nodes. We repeat this selection 20 times for each value $k = 8, 16, 32, 64, 128$ and, for each selection, we analyze (1) for all pairs of ASes, the average and maximal additional path lengths over the k trees, which gives an indication of the traffic overhead, and (2) the number of distinct trees associated with each interface across the network, which gives a high-level indication of the memory complexity.

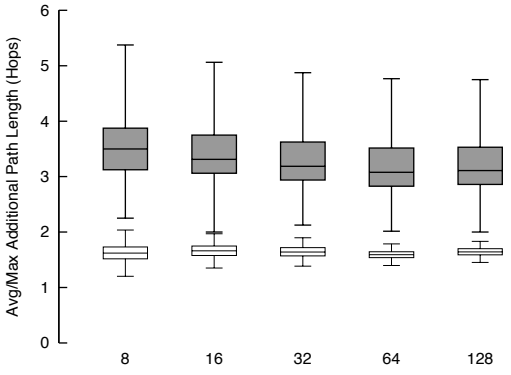

Figure 4: Additional Cost: Average and Maximum

The analysis of the traffic overhead is summarized in Figure 4. For the various sets of $k = 8, 16, 32, 64, 128$ trees, the top and bottom box plots show the distributions of the maximum and average (i.e., expected) additional path lengths, respectively. The box plots extend from the 1^{st} to the 99^{th} percentile. This analysis shows that, in expectation, a k-tree can approximate the optimal routing paths quite well, even with just a few trees. In particular, with only 8 trees, path lengths are extended on average by between 1 and 2 hops and, in the worst case, between about 2.5 and 5.5 hops.

The analysis of the memory complexity is summarized in Figure 5. For the different sizes k of the pool of trees, the chart shows the distribution of distinct trees associated with each interface in all the FIBs across the network from the 1^{st} to the 99^{th} percentile. However, we note that the maximum (in the 100^{th} percentile) extends to much higher values for large sets of trees. In particular, the maximum value is 90 for $k = 128$. Still, this analysis gives very encouraging results, since it essentially shows that, even with many trees, routers must store only a handful of trees per interface. In other words, this analysis shows that trees aggregate well.

4.2 Scalability of Descriptors

The previous analysis gives a high-level indication of the feasibility of a multi-tree scheme by showing that trees in-

[1] http://irl.cs.ucla.edu/topology/, retrieved 29/06/2012.

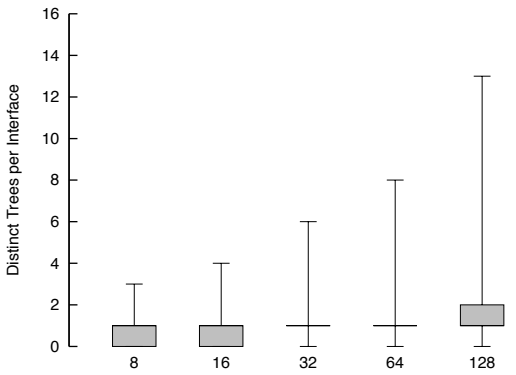

Figure 5: Effective Aggregation of Trees

deed aggregate. However, in order to meaningfully analyze the scalability of the scheme, and also evaluate the feasibility of descriptor-based routing, we must populate the FIBs with plausible sets of descriptors. To that end, we study user behaviors for a number of significant and widely deployed applications. Notice that this analysis is speculative in nature, since the representative applications for which we have significant traces are built for a traditional network interface. Therefore, we must carefully extrapolate plausible future user behaviors over an information-centric network.

4.2.1 Application Workloads

We consider three classes of applications: (1) "pushing" generic Web content and blog posts; (2) "pulling" video content; and (3) "pushing" short messages and following short-message publishers. We now discuss each class of application and the corresponding network workload.

Active Web. We envision a future information-centric network used to actively distribute Web content. So, rather than analyzing traditional Web requests in terms of access to individual servers, we try to understand what users are interested in, which in turn defines the descriptors in subscriptions that would populate FIBs. Since we cannot gain access to comprehensive per-user Web-access logs, we instead infer user interests by analyzing the content that users bookmark. We used the bookmark collection of the Delicious website,[2] which contains the public bookmarks of about 950,000 users retrieved between December 2007 and April 2008 [11]. The data set contains about 132 million bookmarks or 420 million tag assignments posted between September 2003 and December 2007. We assume that users are interested in the content they bookmark, and that they describe the content with the tags they assign. Therefore, we derive plausible subscriptions from user tag sets. We slightly clean the data by applying a simple language-based summarization using stemming and removing duplicate tags. In total we derive 123,248,896 subscriptions for 922,651 users.

We also analyzed data collected from blogs. In particular, we studied the Blog06 collection,[3] which contains 3,215,171 blog posts from 100,649 unique blogs. We use the well-known latent Dirichlet allocation (LDA) algorithm to extract 400 topics that cover these blog posts. We then assume that an author has an active interest in a specific topic if they write more than two relevant posts on that topic. We consider a post to be relevant to a certain topic only if the probability of the post being classified under that topic is more than 20%. For each topic, we select the 10 most relevant tags and use them as a descriptor of the blogger's interests. Ignoring irrelevant posts and users with no significant interest in any topic, we identified 59,185 blogs with 178,189 relevant posts from which we could derive subscriptions.

Video Content. A future information-centric network will facilitate decentralized distribution of video content. In order to determine which content could be offered by users, which in turn determines the descriptors that would populate FIBs, we analyzed data from YouTube. Uploaders of a YouTube video can assign keywords to their videos to allow viewers to find those videos with keyword searches. These keywords were publicly visible until two years ago. In particular, we analyzed a data set derived from 10351 videos published by 782 uploaders in the "Politics" category.

Social Messaging. We analyzed two different aspects of a Twitter data set to generate workloads for a plausible future messaging service. We take into account the structure of the social graph of followers (who follows whom) as well as the content of tweets. Specifically, we assume that followers are generally interested in the messages posted by the authors they follow. We therefore derive plausible subscriptions issued by the followers. We use a graph of 41.7 million Twitter users and 1.47 billion follower relations. For the content we use a collection of 16 million tweets recorded during two weeks in 2011, corresponding to 1% of the total tweets during that period. A Twitter user can attach a number of "hashtags" to each tweet so that other user can issue searches by hashtag. A user can also include links to other content on the Web. Out of the 16 million tweets, we consider those that have both hashtags and links. We collect the hashtags assigned to each link as a descriptor for that link, and then we use these descriptors as the subscriptions for the users who tweeted that link. In total we collected 446,370 subscriptions for 349,753 users.

4.2.2 Assigning Interests to Users

In our analysis we simulate 2.5 million clusters of users to represent a total population of 2.5 billion individual users. We assign user clusters to ASes according to the estimated population of real users for each autonomous system. We then elect among them the users that use each class of application, with an estimated probability derived from the current user population for each of those applications.

For the active Web application, we assume that half the user population is interested in Web content, and adopt the estimate made by NM Incite[4] of 181 million blogs around the world by the end of 2011. Since we did not find data on the actual number of blog readers globally, we used the total number of blogs as the number of users of this service. For the video application, we use the population of 800 million users advertised by YouTube and an estimated 10.9% fraction of uploaders [6]. Finally, for the social messaging application, we adopted the Semiocast estimate of more than 500 million Twitter users in June 2012.[5]

[2] http://delicious.com
[3] http://ir.dcs.gla.ac.uk/test_collections/blog06info.html
[4] http://www.nielsen.com/us/en/newswire/2012/buzz-in-the-blogosphere-millions-more-bloggers-and-blog-readers.html
[5] http://semiocast.com/publications/2012_07_30_Twitter_reaches_half_a_billion_accounts_140m_in_the_US

4.2.3 Simulation Results and Discussion

Table 1 highlights some results of our simulation analysis. We build the FIBs of a gateway router for each of the 42,112 ASes and report the main characteristics of the FIB, including the actual memory required by a concrete implementation of the FIB (emulated). The table shows the results for the most central and, therefore, the *most heavily loaded* router in the network in the presence of a single tree T. We consider all the interfaces that have associated predicates (interfaces that are not on T have no predicates), as well as the largest interface that connects the router to the highest number of final destinations. The FIB represents an original workload of 85 million descriptors for a total of 276 million tags. However, only about 11 million descriptors must be stored in the FIB for a total memory footprint of 518MB. This memory requirement corresponds to an implementation with Bloom filters of 400 bits that, according our rough calculations, guarantee a false-positive probability of less than 10^{-9} for even the largest predicate.

	All Interfaces	Largest
Interfaces	325	1
Destinations	42,112	6,559
Tags	276,501,173	35,814,399
Original Descriptors	85,504,514	10,727,593
Actual Descriptors	10,880,657	1,145,713
Size (MB)	518.83	54.63

Table 1: Worst-Case Memory Requirements for a Single Tree

The results for a single tree are positive. But more importantly, when combined with the analysis of the aggregation of trees in Figure 5, these results suggest that the FIBs of even the most central routers would remain manageable under current hardware technology. In other words, our analysis suggests that routing in an information-centric network can scale, despite or perhaps thanks to the use of expressive descriptors. Our key insight is in fact that user-defined addresses would have no chance of scaling in the absence of intrinsic commonalities, and that such commonalities are more likely to be found in expressive descriptors than in rigid names or, worse, flat labels.

5. CONCLUSION AND FUTURE WORK

To conclude, this paper makes two contributions. First, it proposes a relatively simple and yet effective multi-tree routing scheme to support on-demand content delivery as well as publish/subscribe event notification in an information-centric network with descriptor-based addresses. Second, it attempts to analyze the feasibility of the routing scheme at a global scale using plausible applications for a future information-centric network carefully extrapolated from existing, traditional-network trace data.

One way the scheme and its experimental evaluation should be extended is to study and optimize the use of multiple trees for various network design goals such as maximum throughput and congestion minimization. The problem of decomposing networks into trees for the purpose of routing is well studied in the theoretical literature [9]. However, keeping with the spirit of this work, we want to find practical solutions for the concrete case of Internet topologies and plausible descriptor-based workloads.

Acknowledgments. The work of M. Papalini was sponsored by the Swiss National Science Foundation under grant 200021-132565. The work of A.L. Wolf was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under agreement W911NF-06-3-0001.

6. REFERENCES

[1] A. Carzaniga, M. Papalini, and A. L. Wolf. Content-based publish/subscribe networking and information-centric networking. In *Proceedings of the ACM SIGCOMM Workshop on Information-Centric Networking*, Aug. 2011.

[2] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, Aug. 2001.

[3] A. Carzaniga, M. J. Rutherford, and A. L. Wolf. A routing scheme for content-based networking. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, Mar. 2004.

[4] A. Carzaniga and A. L. Wolf. Content-based networking: A new communication infrastructure. In *NSF Workshop on an Infrastructure for Mobile and Wireless Systems*, number 2538 in Lecture Notes in Computer Science. Springer-Verlag, Oct. 2001.

[5] A. Carzaniga and A. L. Wolf. Forwarding in a content-based network. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, Aug. 2003.

[6] Y. Ding, Y. Du, Y. Hu, Z. Liu, L. Wang, K. Ross, and A. Ghose. Broadcast yourself: Understanding YouTube uploaders. In *Proceedings of the Internet Measurement Conference*, 2011.

[7] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard. Networking named content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 2009.

[8] P. Jokela, A. Zahemszky, C. Esteve Rothenberg, S. Arianfar, and P. Nikander. LIPSIN: Line speed publish/subscribe inter-networking. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*, Aug. 2009.

[9] H. Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, 2008.

[10] D. G. Thaler and M. Handley. On the aggregatability of multicast forwarding state. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, Tel Aviv, Israel, Mar. 2000.

[11] R. Wetzker, C. Zimmermann, and C. Bauckhage. Analyzing social bookmarking systems: A del.icio.us cookbook. In *Proceedings of the ECAI Workshop on Mining Social Data*, July 2008.