



# Scalable Metadata-Directed Search in a Network of Information

Karl PALMSKOG<sup>1</sup>, Alberto GONZALEZ PRIETO<sup>1</sup>, Catalin MEIROSU<sup>2</sup>,  
Rolf STADLER<sup>1</sup>, Mads DAM<sup>1</sup>

<sup>1</sup>*KTH Royal Institute of Technology, Lindstedtsvägen 3, 10044 Stockholm, Sweden*

*Tel: +46 8 790 60 00, Fax: +46 8 790 09 30*

*Email: palmskog@kth.se, gonzalez@s3.kth.se, stadler@ee.kth.se, mfd@kth.se*

<sup>2</sup>*Ericsson AB, Ericsson Research, Färögatan 6, 16480 Stockholm, Sweden*

*Tel: +46 10 7144696, Email: catalin.meirosu@ericsson.com*

**Abstract:** The information-centric paradigm has been recently proposed for the design of future networking systems. A key requirement for realising such systems is having mechanisms that provide efficient, scalable and accurate information search. In this paper, we present solutions for both one-time and continuous searches. Our solution for one-time searches is scalable for its search completion time grows sublinearly with the system size. In addition, the overhead it introduces is evenly distributed. For our solution for continuous searches, we discuss its tradeoff between load (efficiency) and timeliness (accuracy).

**Keywords:** information-centric networking, metadata search, distributed management

## 1. Introduction

Current networks are device-centric in the sense that when a user wants to retrieve a piece of information, she needs to specify the device from where the data has to be retrieved from. Jacobson [1], among others, argues that the design of future networking systems should not focus on devices but on the information itself. In other words, the device-centric communication paradigm should be abandoned for an information-centric paradigm. A key aspect of this new paradigm is that it introduces an information layer, which abstracts users from the devices where the data is stored and the underlying transport technologies.

The 4WARD project [2] is researching this paradigm and (based on it) is designing a new network architecture, called Network of Information (NetInf) [3]. The architecture aims at providing effective and efficient access to information for users across a network.

A key requirement for realising this vision is scalable information search. Scalability is becoming a greater challenge as networks continuously increase in size and they store more and more information. For instance, Google has reached 1 billion ( $10^{12}$ ) unique URLs on the web [4] and is said to maintain a pool of about one million servers [5].

In this paper, we present solutions for efficient, scalable and accurate information search through in-network aggregation, where information is aggregated using spanning trees. Our solutions are based on distributed management algorithms developed following a new approach to network management called In-Network Management (INM). INM is also developed in the 4WARD project [2] and its basic enabling concepts are decentralisation, self-organisation, and autonomy. INM primarily addresses the aspects of the traditional network management paradigm that lead to poor scaling and to slow reactions to changing network conditions [6].

The paper is organised as follows. Section 2 surveys related work. Section 3 presents an overview of the work on information-centric networking in 4WARD. Section 4 presents our solution for information search. Section 5 describes how we have evaluated our solutions using simulations and a prototype implementation. Section 6 concludes the paper.

## 2. Related Work

The literature contains many approaches to problems related to data-oriented network that are built on publish-subscribe architectures. However, the aspect of how to efficiently match potential subscribers to information offered by publishers is left open in many of the proposed solutions. The LIPSIN publish-subscribe system [7] defines a complete network architecture built on a topic-based publish-subscribe mechanism. However, LIPSIN is a forwarding fabric with functionality similar to the underlying Net-Inf machinery. SCRIBE [8] is a decentralized application-level multicast infrastructure based on Pastry. As such, it could well be used as a forwarding fabric for information once a publisher decides to make available information to its subscribers. Siena [9] is a distributed event notification service in which nodes receive messages based on predicates they define over an event schema, allowing for efficient routing of messages from event publishers to subscribers. PIER [10] is an information-centric query engine based on a peer-to-peer architectural substrate using Distributed Hash Tables (DHT). PIER uses hard-coded root identifiers in order to optimize the distribution trees for the queries, and the shape of the tree is dependent on the DHT routing algorithm.

In contrast with PIER, we show how to take advantage of the data-oriented naming implemented by NetInf to build a dynamic in-network search functionality to match potential subscribers to publishers. We allow the subscribers to discover where the content is on the network, and discover where and when new content becomes available on the network (via the continuous search described in Section 4.3), without having to be aware of the existence of the new publishers, or have knowledge of the schema over which Siena selection predicates can be defined.

## 3. Networking of Information

The NetInf architecture proposed by the 4WARD project [3] is based on the information-centric networking paradigm. Similarly to content-centric or data-centric networking [11], this approach enables data to be retrieved without knowledge of the location or device where it is stored. However, in addition, the NetInf architecture permits integrating all kinds of information, including streaming audio and video, as well as store and forward services such as email.

### 3.1 *NetInf Object Model*

A piece of information in NetInf is represented by an Information Object (IO) at the semantic level and, if necessary, is stored as one or more Bit-level Objects (BO). IOs may be structured into hierarchies, e.g. based on their level of abstraction—an IO for a book may be connected to IOs representing different translations of that book. IO metadata can be provided as sets of attribute-value pairs which are IOs in their own right; for example, a document may have *abstract* and *author* as metadata attributes [3, p. 20]. Figure 1 shows another example of a hierarchy of IOs and BOs.

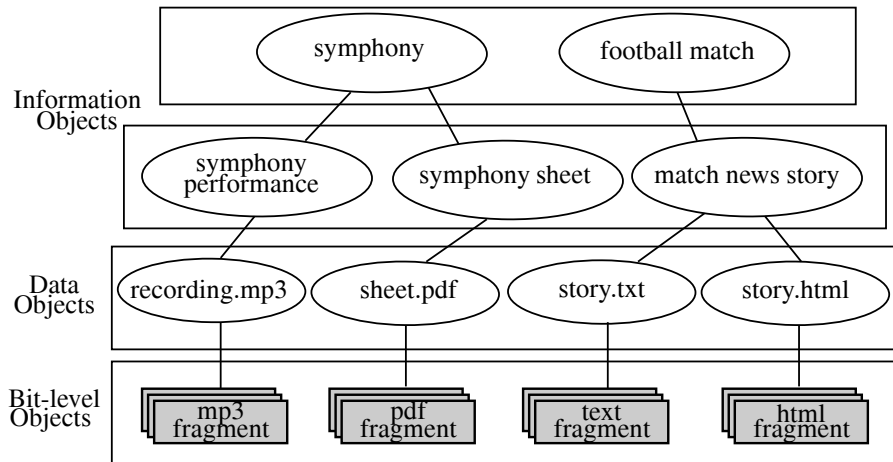


Figure 1: An object hierarchy in NetInf.

### 3.2 Search in NetInf

One central property of NetInf is that a piece of information is not mandated to contain information where its content, i.e. the raw data, is actually stored. Instead, given the identifier of an IO, we can get its locator, whereby the associated BOs can be obtained at will, by using the NetInf name resolution system [3, p. 18]. The recommended approach to designing such a system, which integrates resolution and routing, is to use Multiple-level Distributed Hash Tables (MDHT) [12], implemented on dedicated Dictionary Nodes (DN) in each Autonomous System (AS) of the network [3, pp. 44ff]. However, the identifier of an IO is usually not known in advance by the user when its content is required. Therefore, NetInf must provide advanced search facilities, where users receive lists of IO identifiers that may interest them based on descriptions of metadata [3, ch. 3.3].

Generally, there are two approaches to search functionality. First, there is keyword-based (“textbox”) search, which for a given set of keywords delivers a list of matching IO identifiers sorted after some criteria such as popularity. This kind of search is analogous to what Google provides for today’s Internet [3, p. 25]. Second, there is metadata-directed search, which capitalises on the availability of metadata related to each IO. Given a list of descriptions of metadata, the metadata search function returns IO IDs which match that metadata. Put succinctly, NetInf search “uses the NetInf IO and [Data Object] metadata format with metadata notation convention for finding potential object identifiers” [3, p. 25].

As an example, consider a metadata search related to the hierarchy in Figure 1, where we first search for a symphony performance based on its title and conductor. Then, search results will be a list of IO identifiers for different performances.

### 3.3 Requirements for Metadata Search Solutions

Solutions for metadata search must be efficient, scalable and accurate. Regarding efficiency, we want to be able to find matches among all IOs in the network, but with low overhead and low latency. Since we want to be able to search for IOs across several ASs, the solution must scale with the system size. Specifically, the search latency and overhead must grow sublinearly with the system size. Scalability has proved to be a particularly challenging requirement. Accuracy is also challenging since IOs may be

published and revoked at any time [3, p. 30]. Such changes should be accounted for in our search results. In other words, it is mandatory to perform the search over all IOs existing at the moment the search is requested. Because of this, we cannot adopt existing solutions for data search. For instance, Google performs searches by matching keywords against a vast number of documents of largely unstructured data (i.e. text). The matching process is performed offline, using a (distributed) database of indexed information, which is populated through so-called crawlers that navigate the web on a continuous basis, using links between web pages for this purpose [13]. Clearly, crawlers cannot keep up with the high rate at which IOs are created and therefore the database they populate is never completely up-to-date. This affects the accuracy of the searches.

#### 4. Search Using In-Network Management Algorithms

In this Section, we describe our solutions for efficient, scalable and accurate metadata search. We assume that each DN is capable of performing metadata-based search scoped to the IOs which it holds on behalf of the MDHT.

Our solutions are based on algorithms originally devised for network monitoring. These algorithms realise the INM vision [6], where the main idea is that management tasks are delegated from management stations outside the network to the network itself. The INM approach therefore involves embedding management intelligence in the network, or, in other words, making the network more intelligent. We have decided to adapt these algorithms to perform metadata search since they have proved to be efficient and scalable, as shown in Section 5.

In our solutions, a user search request triggers a global aggregation process where all DNs collaborate to find matching IOs. This means that search will take place concurrently on DNs all over the network. In order to perform the search, our algorithms create a tree that spans all DNs. Partial search results are aggregated bottom-up along the tree, i.e. from child to parent; see Figure 2 for an illustration of the process.

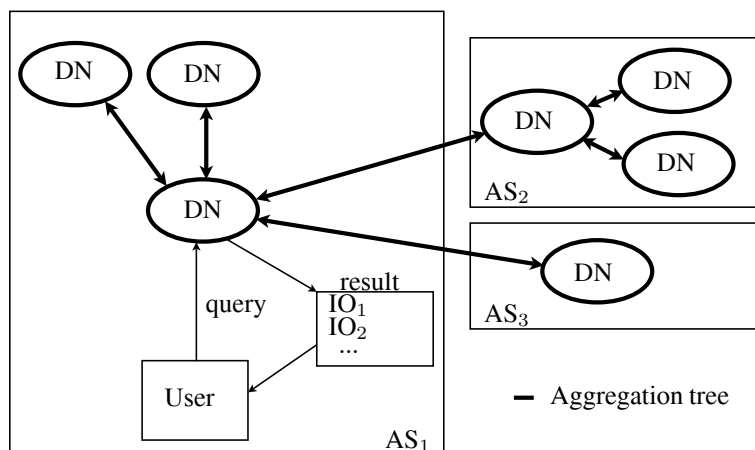


Figure 2: Example of metadata-directed search using tree-based aggregation.

##### 4.1 Tree Overlay Construction

To minimise time complexity when using a DN tree overlay, the root should be close to the user and the aggregation tree should have the Breadth First Search (BFS) property. Since tree construction carries a relatively high cost in terms of time and resources [14],

we use precomputed BFS trees when performing a search operation. In order to avoid having to precompute one tree for each DN in the network, we partition the DN network into disjoint clusters so that the radius of each cluster is at most  $\kappa$ . A larger  $\kappa$  means a longer maximal distance in the network to the aggregation tree root from the user, but also fewer BFS trees to keep track of for DNs.

Distributed graph partitioning can be done in time sublinear in the number of nodes in the overlay [15]. Suppose the resulting number of clusters is  $m$ . The next step is to run a leader election algorithm in each cluster, selecting the DNs which become tree roots. This is followed by the distributed computation of  $m$  BFS trees, which are then stored by each DN in the network. Since we expect churn among DNs to be low [3, p. 42], the DN overlay partition and BFS trees do not have to be computed often. Clearly, repartitioning only needs to be done if a newly added DN has a greater distance than  $\kappa$  to the nearest DN which is the root in a BFS tree, or if a DN crash causes some other DN to have a distance greater than  $\kappa$  to the nearest root.

We expect the parameter  $\kappa$  to be predefined and fixed. However, it can be changed during operation at the cost of repartitioning and recomputing trees, if warranted by changes in the DN topology.

#### 4.2 *One-Time Search Using Echo*

An Echo-based [16], or one-time search, will be performed as follows. Suppose the user sends the query  $q$  to its nearest DN  $d$ , with  $r$  the cluster leader of  $d$  and hence the root of a BFS tree. First,  $d$  will receive  $q$  and pass it on to  $r$ , which initialises the BFS tree  $t$  and sends  $q$  to its immediate children in  $t$ , along with its own identifier to indicate the tree to be used.  $r$  will then search for IOs matching  $q$  among its own IOs. Once  $r$  has received the  $k$  best matches from each of the subtrees rooted in its children in  $t$ , it will send back the  $k$  best matches among all IO identifiers it has received to the user via  $d$  and then terminate the search.

Suppose a leaf non-root cluster DN receives  $q$  and the identifier of  $r$ . It will then retrieve the best  $k$  matches to  $q$  locally, forward them to its parent in  $t$  and stop searching. If a non-root cluster DN that is not a leaf receives the search parameters, it will, similarly to  $r$ , both forward the search parameters to all its children in the tree and initiate a local search, before receiving subtree matches and sending the  $k$  best matches to its parents and terminating the search. The parameter  $k$  is important for scalability; a high value of  $k$  may cause DNs near the root to be overloaded with matches.

Assuming the scenario where a search is made for a symphony based on title and conductor, a one-time search ultimately delivers to the user a snapshot of the available recordings of the symphony at the time the search is performed.

#### 4.3 *Continuous Search Using GAP*

Suppose a user wishes to receive a list of all IOs for news on matches played by her favourite football team, and have it continually updated as more stories appear somewhere in the network. We can perform such a search operation as follows. Suppose again the user's query  $q$  reaches the nearest DN  $d$ , which forwards it to the cluster leader  $r$ . The root  $r$  then starts a continuous aggregation process using the Generic Aggregation Protocol (GAP) [17]. In contrast to an Echo search operation, which will always terminate eventually and notify the user of termination, the user must explicitly end the search interval by sending a message to  $d$ .

In its general form, GAP aggregates in the root the combined weights that are generated at tree nodes by having each node maintain a *neighbourhood table* containing information on status (parent, child, self or peer), tree level and partial aggregates for itself and its tree neighbours. Whenever the neighbourhood table is changed, either by receiving state information from a neighbour or receiving weight locally, all neighbours are sent an update message. In the present case, local weights and partial aggregates are sets of search query matches. Local weights are received by searching directly at the DN tree node. The aggregation function, which defines how partial aggregates from child nodes are added to the parent's partial aggregate, simply performs a union of the matches and then removes all but the  $k$  most relevant matches. A low value of  $k$  allows the user to quickly receive only highly relevant search hits, while a high value offers an exhaustive list of hits and incurs a delay on receiving the most relevant hits.

During a continuous search, it may happen that an IO matching  $q$  that has been previously reported is revoked. If the DN which participates in the aggregation process is notified of this fact, it can remove the IO identifier from its current partial aggregate, and have the revocation be propagated upwards in the tree to the root.

To avoid high load on nodes near the root, GAP imposes an upper bound on message rates along each link in the overlay. This update rate parameter offers a tradeoff between, on one hand, DN load and network congestion, and on the other hand, timeliness of results.

## 5. Evaluation through Simulation and Prototype Implementation

The performance of Echo and GAP has been evaluated experimentally through both simulation and prototype implementation. In [18], we present a model for the performance of Echo that is parametrised by the computational resources at the DNs and the communication bandwidth available. As the model predicts and our simulation results confirm, the time complexity of an Echo search is proportional to the height of the tree of DNs [16, p. 13]. In practice, this means that an Echo operation, which in our case is a complete one-time search, can be performed in around 20 seconds on today's Internet [16, p. 14]. Another attractive property of Echo is that traffic is evenly spread over all network links [16, p. 14]. Compared to other task invocation schemes such as serial or parallel polling, where each network node is contacted by a central node, Echo yields completion times that are several orders of magnitude lower for large networks [19, p. 10]. This is illustrated in Figure 3, where the scalability measure  $S$  is the ratio of the average completion times of a task using parallel polling and our scheme [19, p. 11].

We have implemented a prototype of Echo in a testbed consisting of 16 nodes interconnected via a Cisco Catalyst 2900 switch and performed a set of experiments to validate our performance model [18]. Our results show that the performance predicted by the model is very close to the experimental results. In the experiments we have run, the worst case is an error of 30 ms.

We have evaluated GAP, through simulation [17], in terms of its latency in presenting results. This latency has two sources: the processing time at DNs and the communication delays. In all our experiments, with up to several hundreds of nodes, the latency never exceeds 200 ms. We have also evaluated, through simulation, how GAP adapts to the failure of DNs. If a node fails, the aggregation tree is reconstructed in a sub-second [14]. During this reconstruction the results provided by GAP might

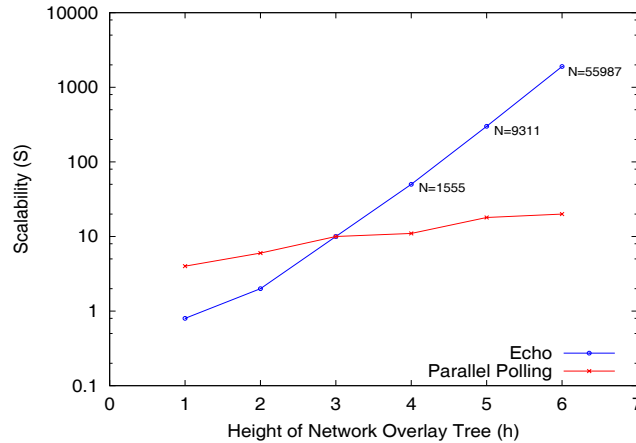


Figure 3: Scalability ( $S$ ) versus network tree height ( $h$ ) for Echo and a parallel polling scheme, with  $N$  the number of Dictionary Nodes.

be inaccurate. However, once the tree is reconstructed, the results of GAP are correct again. Finally, we have implemented a prototype of GAP [20]. Our results show that the behavior of our implementation is very similar to that of GAP running in a simulation environment. This validates our simulation model, demonstrating that its assumptions and simplifications are reasonable.

## 6. Conclusion

In the context of the recently proposed information-centric paradigm, a key building block is efficient, scalable and accurate metadata search. In this paper, we presented solutions for this key block. Specifically, we proposed two solutions based on distributed algorithms. First, we presented a solution for one-time searches based on the Echo algorithm. The algorithm provides a snapshot of the IOs relevant to the search. We have shown that Echo yields completion times that are several orders of magnitude lower than alternative schemes. Second, we presented a solution for continuous queries based on the GAP algorithm. We have discussed the tradeoff in GAP between load (efficiency) and timeliness (accuracy) which is controlled by a GAP parameter (i.e. the update rate). Our future work includes finalising a prototype of our solutions. The prototype is implemented in Java and runs with Linux PCs taking the roles of Dictionary Nodes.

## References

- [1] V. Jacobson, *A new way to look at networking*, 2006 (retrieved December 15, 2009). <http://video.google.com/videoplay?docid=-6972678839686672840>.
- [2] “4WARD: Architecture and design for the future Internet.” Collaborative Research Project within the European Commission 7th Framework Programme (FP7).
- [3] B. Ohlman, editor, “First NetInf architecture description,” tech. rep., 4WARD project, 2009. FP7-ICT-2007-1-216041-4WARD/D6.1.
- [4] J. Alpert and N. Hajaj, “We knew the web was big...,” 2008 (retrieved December 15, 2009). <http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html>.

- [5] Pandia Search Engine News, “Google: one million servers and counting,” 2007 (retrieved December 15, 2009). <http://www.pandia.com/sew/481-gartner.html>.
- [6] G. Nunzi, editor, “In-Network Management concept,” tech. rep., 4WARD Project, 2009. FP7-ICT-2007-1-216041-4WARD/D-4.2.
- [7] P. Jokela, A. Zahemszky, C. Esteve, S. Arianfar, and P. Nikander, “LIPSIN: Line speed publish/subscribe inter-networking,” in *Proceedings of SIGCOMM*, 2009.
- [8] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, “SCRIBE: A large-scale and decentralised application-level multicast infrastructure,” *IEEE Journal on Selected Areas in Communication*, vol. 20, October 2002.
- [9] A. Carzaniga, M. J. Rutherford, and A. L. Wolf, “A routing scheme for content-based networking,” in *Proceedings of IEEE INFOCOM*, 2004.
- [10] R. Huebsch, B. Chun, J. M. Hellerstein, B. T. Loo, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica, and A. R. Yumerefendi, “The architecture of PIER: an internet-scale query processor,” in *Proceedings of CIDR*, January 2005.
- [11] V. Jacobson, M. Mosko, D. Smetters, and J. J. Garcia-Luna-Aceves, “Content-centric networking,” 2007. Whitepaper, Palo Alto Research Center.
- [12] M. D’Ambrosio, P. Fasano, M. Marchisio, V. Vercellone, and M. Ullio, “Providing data dissemination services in the future Internet,” in *Proceedings of IEEE Global Telecommunications Conference*, pp. 1–6, December 2008.
- [13] S. Brin and L. Page, “The anatomy of a large-scale hypertextual web search engine,” *Computer Networks and ISDN Systems*, vol. 30, pp. 107–117, 1998.
- [14] A. Gonzalez Prieto and R. Stadler, “A-GAP: An adaptive protocol for continuous network monitoring with accuracy objectives,” *IEEE Transactions on Network and Service Management*, vol. 4, June 2007.
- [15] B. Derbel, M. Mosbah, and A. Zemmari, “Fast distributed graph partition and application,” in *Proceedings of IEEE IPDPS*, 2006.
- [16] K.-S. Lim and R. Stadler, “A navigation pattern for scalable internet management,” in *Proceedings of IEEE/IFIP IM*, 2001.
- [17] M. Dam and R. Stadler, “A generic protocol for network state aggregation,” in *Proceedings of RVK*, 2005.
- [18] K.-S. Lim and R. Stadler, “Real-time views of network traffic using decentralized management,” in *Proceedings of IFIP/IEEE IM*, May 2005.
- [19] K.-S. Lim and R. Stadler, “Weaver: realizing a scalable management paradigm on commodity routers,” in *Proceedings of IFIP/IEEE IM*, pp. 409–424, March 2003.
- [20] A. Gonzalez Prieto and R. Stadler, “Monitoring flow aggregates with controllable accuracy,” in *Proceedings of IFIP/IEEE MMNS*, 2007.