XTreeNet: Democratic Community Search

Emiran Curtmola
UC San Diego
ecurtmola@cs.ucsd.edu

Alin Deutsch
UC San Diego
deutsch@cs.ucsd.edu

Dionysios Logothetis
UC San Diego
dlogothetis@cs.ucsd.edu

K.K. Ramakrishnan
AT&T Labs Research
kkrama@research.att.com

Divesh Srivastava
AT&T Labs Research
divesh@research.att.com

Kenneth Yocum
UC San Diego
kyocum@cs.ucsd.edu

ABSTRACT

We describe XTreeNet, a distributed query dissemination engine which facilitates democratization of publishing and efficient data search among members of online communities with powerful full-text queries. This demonstration shows XTreeNet in full action. XTreeNet serves as a proof of concept for democratic community search by proposing a distributed novel infrastructure in which data resides only with the publishers owning it. Expressive user queries are disseminated to publishers. Given the virtual nature of the global data collection (e.g., the union of all local data published in the community) our infrastructure efficiently locates the publishers that contain matching documents with a specified query, processes the complex full-text query at the publisher and returns all relevant documents to querier.

1. INTRODUCTION

As the web evolves, we are witnessing a revolutionary process of democratization of information creation in the sense that it is easier to create and publish data on a wide variety of topics; this is evident from the proliferation of blogs, Wikis (e.g., Wikipedia), user-generated content etc. Moreover, it is easier to have the publishers organize in ad-hoc communities based on shared interests; this is true if we consider the popularity of social networking sites like Facebook and Myspace. With the confluence of these trends comes the natural desire to freely exchange data within the community - this includes making one's own data accessible to others within the community, and also be able to query, tag, and comment on the rich global collection that is the union of all local data collections of users in the community. Therefore, a new requirement dimension is the desirability for full-text search evaluation. For instance, a legitimate query on Wikipedia might ask for

> (‡) *all documents that contain terms related to* `Olympics` *and* `Peking` *within a window of 10 words, with* `Olympics` *ordered before* `Peking` [12, 14].

VLDB '08, August 24-30, 2008, Auckland, New Zealand

To fully deliver on the promise of freely exchanging data, any community-supporting infrastructure needs to enforce the key requirement of resistance to censorship by third parties, be they of governmental, corporate or other nature. That is, there should be no easy way for any censoring authority to deny users access to certain publishers and their content. This requirement precludes some proposed approaches that reuse and build on existing centralized technologies (e.g., search engines, hosted online communities, etc.) or decentralized based on pure distributed hash table (DHT) overlays. In the first case, publishers are disintermediated from consumers by the central site in the sense that the central site has control over the visibility of publishers to user queries, and the publishers lose control over who accesses their content. The latter provides a distributed logical abstraction of object identifier lookups over the physical underlay network consisting of simple DHTs (e.g., filename lookups in Napster, Gnutella etc.), individual keyword based lookups such as Minerva [3] or twig queries as in KadoP [1]. While appropriate for P2P search, DHTs provide neither strong censorship resistant systems, nor the ability for complex query resolution for text or XML-based search. For example, a DHT node has complete knowledge of all the publishers that advertise the objects (e.g., keywords) hashed into this node. If the DHT node is compromised then it is possible that information on what each of the managed publishers produced leaks out.

XTreeNet is an architecture based on a novel distributed data index design organized as a union of overlay (i.e., logical) networks, called UQDT. In UQDT, the publishers maintain control over their own content, enabling strong censorship resistance (i.e., stronger than DHT-based systems) among the community members and the published data.

While different queries might hit the same set of nodes, our goal is to balance the community search generated load (number of messages) at nodes during query dissemination while preserving both low index space usage at a node and censorship resistance. We show how to use the UQDT infrastructure to achieve overall load balance and maximize the throughput given a workload of rich-expressive queries (i.e., XQuery Full-Text queries) through efficient query routing algorithms and optimization strategies over UQDT.

Moreover, XTreeNet allows users to tune the system parameters (i.e., the number of overlays, the overlay topology, the query routing strategy) to get insight into the existing design tradeoffs and community search process. Although one of the drivers for our architectural design concerns censorship resistance, this is not the main focus of the demo.

Because of the increasing number of online communities and social network sites, the need for powerful search expressivity (beyond simple keyword search) and the need for democratic information exchange by protecting the dissemination infrastructure against censorship, we believe a demonstration of our UQDT infrastructure design and proposed techniques will be of general interest. The demo is available at the following location [1].

We recall next the UQDT index infrastructure and the distributed query dissemination techniques from [7]. Then, we describe our network system organization that supports efficient multi-overlays (i.e., logical) construction and we detail the XTreeNet demonstration scenarios.

2. DISTRIBUTED QUERY PROCESSING

Given a user query, XTreeNet identifies the relevant data sources that contain matching documents and returns them to the querier. In a first step, the system identifies only the sources of interest and contacts them based on a simplified form of the query (i.e., the set of keywords in the query). Then, the sources decide based on their own access policy (i.e., the querier's identity or the history of who accessed what) whether to release the matching documents to the queriers by running the full query on their corresponding local document collection. We detail these features next.

2.1 Efficient Data Source Discovery

Our indexing solution targets a service-oriented logical network infrastructure, in which we distinguish two types of nodes. There are data *publisher nodes* (the community members) that provide data services and connect to the network via direct links to nodes at its edge. The data are indexed inside the network, which consists of a set of inter-connected and reconfigurable *router nodes*. These are responsible for routing queries to the relevant publishers.

XTreeNet is based on on a novel distributed data index design, called UQDT, that is organized as a union of query dissemination trees (QDTs) realized as an overlay (i.e., logical) network. QDT's purpose is to focus the query forwarding with high probability towards only a subset of all publisher nodes that contain matching documents. Regardless of which querier initiates a query Q, Q is sent to the QDT's root, and it propagates down the tree to the publishers.

For the purpose of information discovery and flexible querying, documents and queries are represented as collections of *content descriptors*, called CDs. A CD is an abstraction of a hierarchical data item. In our case, CDs are keywords in a path context. When the context is empty this corresponds to simple full-text keyword search. The CD collection that describes the user query and the documents are automatically extracted.

We adopt a data partitioning approach in which we partition the global CD collection into multiple partition blocks. Each block is associated with a QDT which takes care of forwarding the data that falls in this partition block. We build smart hierarchical summaries at each node of a QDT to facilitate early pruning by disseminating Q only to the relevant publishers. Each node summary consists of the union of all advertised CDs by publishers in its subtrees for the corresponding QDT. We implement node summaries as counting bloom filters for their well known properties: compactness and quick probabilistic set membership of CDs.

[1] http://db.ucsd.edu/xtreenet

We employ the following query routing algorithm. Note that the data partitioning scheme determines also a partitioning among the CDs of a query. Therefore the query can be disseminated on all or either of the QDTs that map to the query partitioning blocks. We show in [7] that it suffices to send the query to the root of only one QDT while still preserving the query semantics. When a router node receives the query message, it forwards it in parallel to each of its children in QDT if and only if the CD set of the query is contained in the node summary. When a publisher node is reached, the full query (not just the CDs extracted from the query) is evaluated on the local database. Any matching documents are sent back to the querier. The advantage of this approach is that the query dissemination is very selective since pruning decision at each node is done on a conjunctive basis (conjunction of CDs).

The number of QDTs to setup in the network and the QDT to send queries are optimization problems for throughput maximization that we solved in [7]. The intuition behind our decision for the number of QDTs is that since the load in one QDT decreases from the root to leaves for any query workload, we try to balance the load of each router node by assigning them to different levels across the different QDTs. One solution which behaves well in practice is to cyclically permute the nodes on tree levels across all QDTs such that all routers appear precisely once in the top levels of any QDT. Picking the QDT to disseminate the query is based on query selectivity estimation techniques. In particular, we avoid routing based on query blocks that contain popular CDs. In [7] we show that keeping track of a very small number of advertised popular CDs (2-3%), which we call partially informed strategies, XTreeNet achieves almost as good performance as if we had the selectivity information for all the data collection, called the fully informed routing strategy. Note that the latter approach is infeasible in a complete decentralized setting as ours.

2.2 Query Evaluation at Source

Without loss of generality, we employ XML data sources. Users publish and query XML repositories. A CD is then the abstraction of a keyword in an XPath context. To be able to express powerful queries over such data we adopt the XQuery Full-Text [14] (XQFT) standard specification which permits composable full-text search primitives such as simple keyword search, Boolean queries, and keyword-distance predicates over XML data. In order to process such expressive class of queries we deploy an XQFT engine processor at each source. In particular, we leverage the GalaTex [6] platform processor together with the optimization framework developed in [2].

The query semantics is to iterate through all the documents at all the identified sources and return to the querier only those matching the full XQFT query. For efficiency, the backend text engine is build on top of a local index store that contains posting lists for documents at the publisher.

2.3 Example

We show in this section an example of distributed query processing in the UQDT infrastructure. The same query (‡) from Section 1 can be expressed in XQFT as follows:

```
doc/title[.  ftcontains ‘‘Olympics’’ ftand
‘‘Peking’’ window 10 words ordered]
```

The corresponding CDs are cd_1=doc/title/"Olympics" and cd_2=doc/title/"Peking". Let us assume we use a data par-

titioning scheme that splits the two CDs into different query blocks. Each query block is associated with a different logical QDT overlay. Our routing strategy chooses to route the query based on the most selective of the CDs (consider cd_1 is more selective than cd_2) therefore, the query is sent to the QDT corresponding to cd_1. The dissemination on this particular QDT as well as the QDT's internal organization is shown in Figure 1 given that cd_1 appears in the node summaries for nodes 4, 6, 10, 20, 23, 1 and 2 and being actually advertised only by publishers P_2 and P_3. Note that P_2 and P_3 is a superset of the publishers that can actually return matching documents. When reaching the candidate publisher nodes (in this case the leaf nodes P_2 and P_3) the full XQFT query is evaluated locally using GalaTex and the matching documents are returned to the querier.

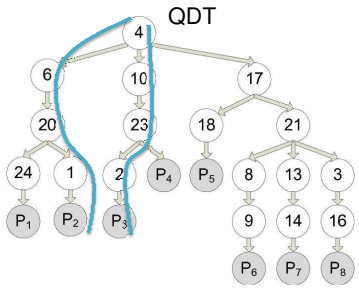

Figure 1: Routing cd_1=doc/title/"Olympics" on the corresponding QDT.

3. DEMONSTRATION SCENARIO

We propose to demonstrate interactively all the functionality of XTreeNet. The purpose of this demo is a proof-of-concept to show the new architectural design in action as well as the flexibility and efficiency of distributed query processing; in particular, the efficiency in identifying relevant sources and XQFT query processing at the source.

To obtain a true-to-life community, we consider a distributed community that shares a real data collection, namely an XML dump of Wikipedia, comprising about 1.1 million real Wikipedia documents which amount to 8.6 GB [8].

To facilitate the peers interconnectivity in the community we use a 3^{rd} party set of logical routers. We can imagine the alternative solution as well, in which each peer publishes data, queries and forwards queries.

We describe next the underlying network infrastructure and the overlay network support. We then present the demonstrated features.

3.1 Network Infrastructure

The XTreeNet distributed engine relies on an overlay network to manage the distributed index across the XTreeNet participants. The index consists of multiple, custom designed QDTs, where each node contains a counting bloom filter. Each filter continuously updates the index by processing the stream of updates from its child nodes. Because a bloom filter is an in-network *aggregate*, it may be computed efficiently across these trees. However, XTreeNet's wide-area deployment requires the maintenance of the bloom filters to be scalable, network-aware and failure resilient. To achieve these operating criteria, XTreeNet uses the Mortar [11] stream processing engine, a platform for instrumenting endhosts with stream operators.

Mortar provides a number of features that are important for XTreeNet. First, it supports user-defined in-network aggregates, allowing XTreeNet to create hierarchical index summaries by simply extending Mortar with custom stream operators that implement the spectral bloom filter. Each operator indexes local content and propagates index changes to its parent. Mortar also arranges the nodes in the tree in a network-aware fashion, ensuring that the majority of participants are within a low-latency horizon of the root. Further, unlike DHT-based in-network aggregation systems, Mortar allows applications to control the design of additional trees. This is critical, as XTreeNet balances the query load by controlling the shape of the set of trees implementing the index. Other features provide fast and reliable operator management (installation/removal) and accurate stream processing in the presence of node failures and unsynchronized clocks.

We evaluate XTreeNet by deploying 1,000 to 10,000 XTreeNet peers over an emulated network using the ModelNet [13] emulator. Modelnet is a large-scale evaluation environment that combines the realism of an Internet testbed with the ability to execute experiments in a reproducible fashion. In Modelnet, unmodified applications run over unmodified operating systems and network stacks, while the emulator subjects the application's traffic to the bandwidth, delay and loss constraints of the emulated network topology. In our setting, 34 physical machines, running Linux 2.6.9 and connected on a Gigabit network, emulate an Internet-like topology built with the Inet [10] topology generator.

3.2 Running Scenario

The XTreeNet query interface at a peer is very simple, yet it hides a powerful and efficient architectural design. The web browser interface accepts keyword search queries in the XQuery Full-Text style allowing for a multitude range of queries supporting from simple keyword queries, keywords in a context (XPath) to complex queries with predicates on the keyword positions (e.g., proximity distance predicates).

In addition to the query input, the interface allows a user to tune the various system parameters: setting the number of QDTs in the UQDT structure, the QDT to route a query on, or the routing state amount maintained at each node to guide the routing process.

The result of executing a query is a page containing links to the matching documents together with a detailed section containing statistical and routing information decisions. Such information includes the suggested routing QDT, both the processing and the forwarding loads generated in the system as the number of total exchanged messages and a break down on execution times for the various stages of query processing. We enumerate in the following various demo scenarios of interest.

Query Routing and Processing in XTreeNet. First, we show the interactive functionality to search the global community data collection by executing different ad-hoc XQFT queries. We demonstrate the use of UQDT as a distributed index infrastructure that provides support for complex querying via multiple index lookups during query forwarding and then running an XQFT processor at the publisher.

After a peer issues a query, we present a visualization of the internal routing flow of the query into the network. First, the system decomposes the query into query blocks based on the UQDT partitioning scheme over the keywords. By

using these query blocks, we show how the system smartly decides the corresponding QDT overlay where the query is actually routed on. We show next the routing path of the query on that QDT by specifying the nodes and the links touched. At the end of routing, the reaching leaves are the candidate publishers. Let us notice that this is a super set of relevant publishers since they have been selected based on the conjunctive part of the query.

Finally, the full query is tested at these publishers by running an XQFT processor to check and retrieve the actual matching documents. The querier has the option of interactively changing the set of matching documents, simply by varying the keyword conditions in the query (e.g., adding or removing keywords and keyword predicates).

Democratization of Publishing. The ability for an individual publisher to dynamically control the access to the content she owns, including the ability to make visible what information she wants to, on a selective basis to different users, is a highly desirable aspect in a democratic network.

A key aspect of such an information access infrastructure, that is a key to the democratization of the Internet, is to make all requests for information (queries) available to all the publishers who may have relevant information and allowing them to determine their response to the query.

We show how publishers can maintain control over their own data. Depending on what they decide to advertise, independent on the actual local published content, different queries may reach them. In this demo we do not enforce a particular publisher policy to access their data even though one can think of sophisticated ways of doing it. We consider, for now, that publishers answer all received queries correctly.

Moreover, we do not consider for the scope of this paper the system's robustness to failures, as the main focus is to show how to leverage the new UQDT index structure for efficient distributed query processing and privacy in P2P publishing. We defer this as future work. However, preliminary analysis shows that this is feasible based on the Mortar infrastructure (as described in Section 3.1).

Our censorship-resistant UQDT infrastructure prevents leaking any information about which publishers are capable of answering a given query. In the case of compromised nodes, we allow to zoom into individual routers and introspect the actual information that is being kept as part of the UQDT index. We show that getting hands on the local bloom filter and the overlay connections at a router does not reveal much information.

Interactive Tuning. As we mentioned previously, we let the user interact with the main parameters of the system in order to get a feeling of the various tradeoffs.

For instance, the querier can choose to disseminate queries over specific QDT overlays. When a peer issues a query, the system suggests what would be the best query routing strategy based on techniques described in Section 2.1. However, the querier can send the query to any of the existing QDT overlays for the following reasons: she may have domain or external knowledge on the selectivity of the overall published data items, or may want to analyze generated traffic on different QDT dissemination, or just has a preference for a particular set of nodes.

In the case that the querier decides not to send the query to the suggested QDT, we run the query on both variants (on the suggested QDT as well as on the querier-chosen QDT) and even though the number of answers is the same, we show a comparison of the amount of generated traffic.

Similarly, the querier may change the amount of routing state. This can lead to changes in the suggested QDT among the available QDT overlays. Intuitively, the more state is maintained at a node, the more precise the suggested QDT is relative to the best routing (e.g., when all the selectivity state of published data items is known).

Balancing the load. In this scenario, we show that XTreeNet can be used to balance the overall traffic. The load is near-optimum uniformly distributed among the peers.

We run a query workload based on a uniform distribution of queries with the number of conjuncts varying from 1 to 10. We stress the load such that each conjunctive query has a match in the global data collection. We let the user dynamically setup the UQDT configuration by picking the number of QDTs. For choosing the overlay topology we employ off-the-shelf tools developed by network research for multicasting (e.g., generated by SCRIBE [5]).

During the query workload execution, we collect statistical information. At the end, we report processing and forwarding load histograms between different number of QDTs. We show that for 15-QDTs the load is well balanced in the system and therefore, the overall throughput is maximized. We define the throughput as the number of queries answered per unit of time. At the same time, in the 1-QDT case this shows the high congestion in the system.

4. CONCLUSION

In this paper we presented XTreeNet, an efficient infrastructure that empowers information publishers to join censorship-resistant communities and query their global data collection in an ad-hoc fashion using expressive queries.

5. REFERENCES

[1] S. Abiteboul, I. Dar, R. Pop, G. Vasile, D. Vodislav, and N. Preda. Large scale P2P distribution of open-source software. In *VLDB*, 2007 Demo.

[2] S. Amer-Yahia, E. Curtmola, and A. Deutsch. Flexible and Efficient XML Search with Complex Full-Text Predicates. In *SIGMOD*, 2006.

[3] M. Bender, S. Michel, P. Triantafillou, G. Weikum, and C. Zimmer. Minerva: Collaborative P2P Search. In *VLDB*, 2005.

[4] B.H. Bloom. Space/time trade-offs in hash coding with allowable errors. Commun. ACM, v.13 n.7, July 1970.

[5] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC)*, 2002.

[6] E. Curtmola, S. Amer-Yahia, P. Brown, and M. Fernandez. GalaTex: A Conformant Implementation of the XQuery Full-Text Language. In *XIME-P*, 2005.

[7] E. Curtmola, A. Deutsch, K.K. Ramakrishnan, and D. Srivastava. Censorship-resistant Publishing. In *Technical Report CS2008-0919, UC San Diego*, March 2008. http://db.ucsd.edu/xtreenet/xtreenetTR08.pdf.

[8] L. Denoyer and P. Gallinari. The Wikipedia XML Corpus. In *SIGIR*, 2006.

[9] L. Fan, P. Cao, J. Almeida, and A. Broder. Summary Cache: A Scalable Wide-Area Web Cache sharing Protocol. In *IEEE/ACM Transactions on Networking*, 8(3), 2000.

[10] Inet. http://topology.eecs.umich.edu/inet

[11] D. Logothetis and K. Yocum. Wide-Scale Data Stream Management. In *Usenix Annual Technical Conference*, 2008.

[12] A. Trotman and B. Sigurbjrnsson. NEXI, Now and Next. In *INEX*, 2004.

[13] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker. Scalability and accuracy in a large-scale network emulator. In *Proc. of OSDI*, 2002.

[14] The World Wide Web Consortium. XQuery 1.0 and XPath 2.0 Full-Text 1.0. http://www.w3.org/TR/xquery-full-text/.