

A Bandwidth Model for Internet Search

Axel Uhl

Interactive Objects Software GmbH
Basler Str. 65
79100 Freiburg
Germany
axel.uhl@io-software.com

Abstract

In this paper a formal model for the domain of Internet search is presented that makes it possible to quantify the relations between important parameters of a distributed search architecture. Among these are physical network parameters, query frequency, required currency of search results, change rate of the data to be searched, logical network topology, and total bandwidth consumption for answering one query. The model is then used to compute many important relations between the various parameters. The results can be used to quantitatively assess, streamline, and optimize distributed Internet search architectures.

The results back the general perception that a centralized approach to Internet-scale search will no longer be able to provide the desired coverage and currency, especially given that the Internet's content keeps growing much faster than the bandwidth available to index it. Using a hierarchical distribution approach and using change-based update notifications instead of polling for changes allows to address sets of objects that are several orders of magnitude larger than what is possible with a centralized approach. Yet, using such an approach does not significantly increase the total bandwidth required for a single query per object reached by the search.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, requires a fee and/or special permission from the Endowment.

**Proceedings of the 28th VLDB Conference,
Hong Kong, China, 2002**

1 Introduction

Internet search as being done today suffers from bandwidth limitations. Major Internet search engines compute and maintain a central index of hundreds of millions of documents. But since content grows much faster than the bandwidth available for indexing it, this approach increasingly falls behind in its effort to index a reasonably large share of all accessible information [1]. New approaches based on hierarchical distribution of index data and smart, incremental, change-driven updates to this data have to be implemented in order to make Internet search scalable.

However, this perception carried several uncertainties. Open questions were for example how the total bandwidth use would increase if instead of central indexing a distributed infrastructure were used. How would the execution time of queries be affected by such a distributed approach? How would the layout of the logical network topology affect the ratio between the number of reached objects and the bandwidth required to answer a single query over this number of objects? How much bandwidth could be saved if search engines did not have to poll web pages without knowing whether or not they changed?

These questions show that there is a need for a formal model that will provide quantitative answers. This paper presents such a model for the domain of Internet search. It captures, among others, bandwidth consumption of all related network activities, query execution time, query frequency, logical network topology, physical network parameters, search result currency, as well as the change rate of searchable data.

Using the model, several important relations between the various model parameters can mathematically be proven, for example the dependency of the logical network topology on the physical network parameters and the query execution time; or the dependency of the query execution time on the number of objects to be covered by the search.

This enables streamlining and optimizing the search

architecture along given user and market preferences, while being able to quantify the effects up-front.

The model is then evaluated by using it to visualize the impact of the various parameters of the network topology on bandwidth efficiency, number of searchable objects, and query execution time. It is also used to compute the bandwidth impacts of maintaining forward knowledge like keyword indices using polling instead of receiving updates upon change only.

1.1 Related Work

In [2] a good overview of the parameters that influence Internet search can be found. Bandwidth is considered mainly in the context for optimizing updates to forward knowledge. It does, though, lack a set of other important parameters like, e.g., currency and does not talk about bandwidth-efficient distributed storage of forward knowledge.

A pragmatic, yet powerful model for formalizing distributed systems regarding, among other things, their bandwidth constraints has been provided in the *LogP*-model [3]. The definition of *latency* is used here as well and denotes the time between the start of sending and receiving a message. The *LogP* model uses the notion of a *gap* between network transmissions, thus defining the speed with which data can travel over a network connection. Here, the more intuitive concept of *bandwidth* shall be used instead. Furthermore, computing time, as defined in the *LogP*-model by the parameters o (the *computational overhead* for sending a message over the network) and P (the number of processors in a node), is not considered here because it is deemed an insignificant constraint as compared to bandwidth and latency limitations for the problem of Internet search.

A good overview of different efficient communication patterns in so-called *star trees* can be found in [4]. Unfortunately, network parameters like latency or bandwidth are not considered in finding the optimal tree structure. The article introduces the term *message complexity* meaning the number of messages that have to be sent in order to transport the original message to all recipients. This is an important measure when bandwidth consumption comes at a cost and is also used in this paper.

[5] introduces *arrangement graphs* as a solution to span a number of network nodes with a tree that then allows optimal broadcasts to the set of spanned nodes. There are two problems with this approach: It does not allow for nodes dedicated only to forwarding messages and not acting as final receiver; and it does not consider network parameters for the graph optimization.

A technical infrastructure tackling the problem of Internet search has been presented in [6, 7]. It suggests a hierarchical layout, allowing for intermediate tree nodes dedicated only to query forwarding and re-

sult merging. The formalization developed here can be applied, e.g., to systems built with that infrastructure.

1.2 Terminology

The *latency* between two network nodes is the time that passes between the start of sending a message and the start of receiving that message. This definition resembles that in the *LogP*-model [3]. Latency is independent of message length.

Bandwidth identifies the speed with which data can be sent across network connections. It is defined as the number of bytes that can pass through a connection per second. Note, that bandwidth is defined independently of latency. If W is the bandwidth of a connection, L the latency of that connection, and Q the size of a message to transfer, then the total time between the start of sending the message and the end of receiving the message is $L + \frac{Q}{W}$ (see also figure 2).

A *level* in the tree are all tree nodes that have equal distance to the tree's root node. This distance at the same time is used as the level's number. Thus, the root node is on level 0, the children of the root node are on level 1, and so on (see figure 1).

The terms *result object* and *response object* are used synonymously and identify the type of object that is returned as answer to a query.

The term *forward knowledge* is used in the definition given in [8]. Forward knowledge contains data about a searchable source. It can be used to answer a query or at least to make good query routing decisions. It can be moved across the network and can be stored persistently. Forward knowledge can have a *lack of currency*, which is defined to be the time between the last change in the underlying data that would have caused a change in the forward knowledge that has not yet been updated and the current time (see also figure 7). A possible implementation of forward knowledge can, e.g., be an inversed keyword index.

1.3 Organization of this Paper

Section 2 introduces a tree structure as the distribution model for searching, defines the behavior of each node in the tree, and formalizes this model with a set of parameters. Section 3 uses this model to compute the tree shape based on the maximum allowed query execution time, the network parameters, the query frequency, and the query and response object sizes. The shape is defined by the number of children that each node in the tree has.

Section 4 considers the bandwidth required for maintaining forward knowledge about a data source. Here, parameters like object change rate and desired currency play an important role. The results are combined with the results on the tree structure from section 3. In section 5 the results are used to compare today's prevailing central-index based approach with

a distributed approach. Section 6 concludes the paper and provides topics for further research.

2 Model for the Domain of Internet Search

The model considered here consists of a hierarchy of network nodes. The number of children that a node t has is termed $S(t)$. Each node either acts only as a forwarder of queries (referred to as a *trader*), or it contains actual searchable data from which it can instantly answer a query. Nodes of the latter type are also referred to as *searchable* and are the producers of result objects for a given query. The searchable data in the leaves of the hierarchy may represent forward knowledge collected from various other distributed resources.

Initially, the model is restricted to a directed tree. An example is illustrated in figure 1. Later, in subsection 3.4, this model will be extended to allow replication of nodes in the hierarchy, turning the tree into a more general directed acyclic graph (*DAG*). Thus, a node in the extended model can have more than one parent node.

The dynamics of a trader node are defined as follows: A query that is received is forwarded to all child nodes, one after the other. Serializing the messages rather than interleaving them maximizes the time each child node has for computing its results while leaving the time and bandwidth spent at the trader node for forwarding the queries unchanged. The size of such a query object is termed Q . Each child node returns the best match for the query as a response object of size R , and the trader selects the best match of those returned by its children and returns it to its parent or the client who sent a request to the root trader.

For simplicity it is for now assumed that each tree node produces exactly one best result. When the client asks for the second-best match the trader will retrieve the second-best match from the child node from which the previous best match was returned and compares it against the already obtained results from the other children. Again, the best match is returned to the parent / client. Further research has to show what the optimal number of results is that each node should return, given the network parameters and the total number of desired results. Using this “streaming technique” for the results it is guaranteed that for retrieving the best match each node has to return exactly one result, namely its best match regarding the query, thus keeping the bandwidth consumption constant regardless of the number of total matches found in the whole hierarchy.

If a leaf node s maintains forward knowledge like a keyword index of one or more searchable data sources it will require communication with those sources in order to keep its data current. It is assumed that sources will notify the leaves upon changes that would affect

the forward knowledge¹. A source in this model is considered as consisting of a number of *retrievable objects* where each of those can be the basis for a result object. An example for a retrievable object is an HTML document residing on a web server. The total number of retrievable objects united in s 's forward knowledge is identified by $D(s)$.

A leaf node may tolerate a certain lack of currency, i.e. it may choose to save bandwidth at the cost of decreased currency of its forward knowledge. It splits its bandwidth use into receiving queries and sending out responses, and updating the forward knowledge it maintains.

It is furthermore assumed that answering a query from forward knowledge stored in a leaf node takes no measurable time. This assumption is backed by two thoughts: forward knowledge lookups typically are performed in time complexity $O(1)$ by using indexing and hashing techniques; and CPU power can much easier be scaled up for this particular problem compared to network bandwidth, also regarding cost.

Note, that with this model a central-index search engine architecture can be described as well as a completely decentralized approach as implemented by *Gnutella* (see e.g. <http://www.clip2.com/gnutella.html> or <http://www.tch.org/gnutella.html>). The central approach is described by only one searchable node and no traders, with the searchable node keeping the forward knowledge of all sources to be searched with the system. The completely decentralized approach is modelled from the client's perspective as one trader with a very large set of child nodes, each being searchable.

Both solutions, the central as well as the highly distributed, have their well-known drawbacks and bottlenecks. The presented model will help to understand the space in between these two extreme models better and grasp them quantitatively regarding their query-response-time behavior and their bandwidth requirements.

The described model is now formalized by defining a set of parameters that capture the essential characteristics of a particular instance of the model. Table 1 shows the parameter definitions.

Figure 1 shows a simplified example instance of the model, omitting some parameters like the query frequency, bandwidth or currency of forward knowledge.

3 Computing the Tree Shape

Based on the model defined in the previous section, these definitions will now be used to compute the shape of the tree of traders and searchables that maximizes the number of reachable searchables, based on the bandwidth B , the size of the query and response

¹Note, that this assumption is currently *not* fulfilled by standard web server technology. It would, however, be simple to implement, as shown, e.g., in [6].

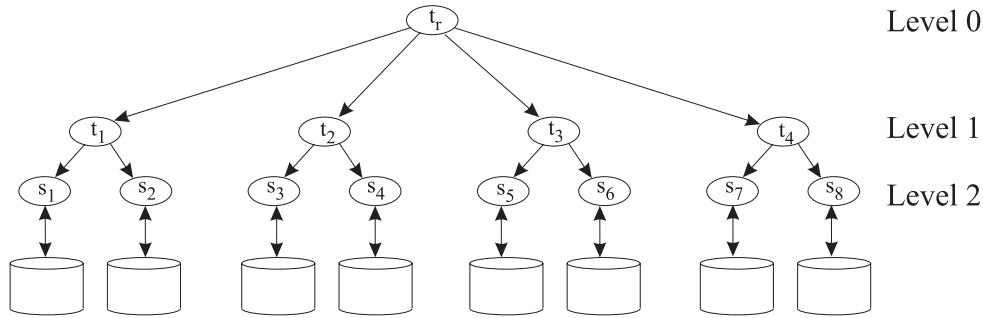


Figure 1: Example for a model instance. Three levels of nodes with the leaves accessing data sources in order to maintain forward knowledge about them. $S(t_r) = 4$, $S(1) = 2$, $O(t_r) = 8$. The datasources at the bottom contain $D(s_1), \dots, D(s_8)$ retrievable objects. The total number of retrievable objects thus is $\sum_{i=1}^8 D(s_i)$.

Table 1: Model parameters

name	unit	description
$S(t)$		number of children of trader t . Instead of t the hierarchy level j may also be used as parameter in case all traders on the same level have the same $S(t)$. The <i>level</i> of a node is its distance to the root node. The root is on level 0, and the root's children are on level 1.
$q(t)$	<i>sec</i>	time that trader t has after having received a query until it has to start sending back the result for that query. Instead of t the hierarchy level j may also be used as parameter in case all traders on the same level have the same $q(t)$.
$D(s)$		number of retrievable objects about which a searchable node s maintains forward knowledge. If used without the argument s , then D identifies the average number of retrievable objects per searchable node.
$O(t)$		number of leaf tree nodes that have t as their direct or indirect parent
$A(s)$	<i>bytes</i>	bytes to transfer to update the forward knowledge for one retrievable object from a searchable source s . Note, that through synergetic effects when concurrently updating the index for multiple retrievable objects the average size per object will typically be lower than the size for updating the index for only one. So $A(s)$ can be seen as upper bound.
$C(s)$	<i>sec</i>	average change rate for each retrievable object of searchable source s regarding its query response behavior (and thus its index representation). Measured as the time between query-relevant changes. This means, every $\frac{C(s)}{D(s)}$ seconds on average there will occur a query-relevant change in Searchable s .
F	sec^{-1}	number of queries per time at a trader or searchable node
Q	<i>bytes</i>	size of a query object
R	<i>bytes</i>	size of a result object
$Y(s)$	<i>sec</i>	lack of currency of the forward knowledge at searchable s ; measured as the maximum passed time since the first query-relevant change of the data on which the forward knowledge is based that occurred after last updating the forward knowledge and now (see figure 7). With this definition a searchable s 's forward knowledge is current with $Y(s) = 0$ (no relevant change in the data since last update).
W	$\frac{bytes}{sec}$	bandwidth available between two network nodes. Note, that this assumes that all network nodes have the same bandwidth at their disposal and that bandwidth is identical for sending and receiving data.
L	<i>sec</i>	Latency between two network nodes. Note, that this assumes that the network latency is identical along all network connections, and that it is identical for sending and receiving data.

objects Q and R , the query frequency F , and the time $q(t_r)$ that the user allows the root trader t_r to use to answer the query. The tree shape is defined by the function $S(t)$ which tells the number of children for a tree node t .

The two fundamental ideas in solving $S(t)$ based on the input parameters W , Q , R , F , and $q(t_r)$ are that

- there is a direct connection between $q(t)$ and $S(t)$ given the input parameters. This connection is defined by the time it takes node t to forward the query of size Q to $S(t)$ child nodes and receive the results from the same number of children.
- the time $q(t_i)$ that one of t 's child nodes t_i has to answer a query follows from the number of t 's children $S(t)$.

With this, $S(t_i)$ for t 's child nodes t_i can be defined in terms of $S(t)$, leading to a recurrence relation for S . It will turn out that by assuming $R \approx Q$ this recurrence will furthermore be independent of the child index i and can then be solved with the given input parameters.

It will then be proven that $S(t)$ yields identical values for all t on the same tree level under the assumption $R \approx Q$ and that $S(j)$ with j identifying the tree level is a monotonically decreasing function in j with a negative limit. This implies that the tree has a finite depth, as there is a level j_{\max} on which the tree nodes have no children. Intuitively, this is the level on which the tree nodes are not given enough time to forward the query to any other tree nodes.

Finally, this section will consider high query frequencies F leading to bottlenecks in the top of the tree, how they affect the tree shape, and how these bottlenecks can be eliminated.

3.1 Computing $S(t)$

If $S(t)$ is the number of children of trader t , then the child at position $1 \leq i \leq S(t)$ has time

$$q(t_i) = (S(t) - i) \frac{Q}{W} + (i - 1) \frac{R}{W} - 2L \quad (1)$$

between having received the query and sending out the result (see also figure 2). This is the time the trader t has to spend to send the queries to all remaining children and receive the results from all previous children, minus the network latency for the sending and the receiving direction ($2L$).

The dynamics of the model as described in section 2 immediately imply that the runtime of a query at node t is²

$$q(t) = S(t) \frac{Q + R}{W} \quad (2)$$

²Note, that the number of children that is represented by $S(t)$ is an integer value. In actuality, the equation would have to be put as an *approximation*, and the number of children results from applying the *floor*-function to $S(t)$.

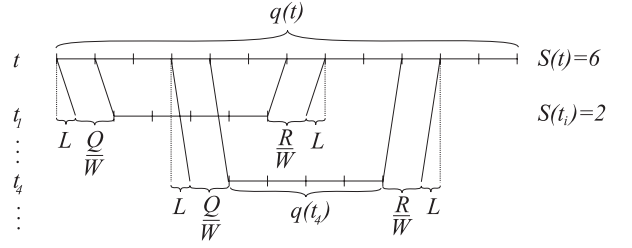


Figure 2: Relations between $q(t)$, $q(t_i)$, $S(t)$, bandwidth W , query size Q , response size R , and latency L . The first half of the processing time is spent forwarding the query, the second half is spent receiving the responses.

with $S(t)$ being the number of children that node t has to forward the query to. This provides the mapping between $S(t)$ and its query execution time $q(t)$.

Let t be an arbitrary trader node, and let $t_1, \dots, t_{S(t)}$ be t 's children. Applying (2) for t_i and solving for $S(t_i)$ yields $S(t_i) = q(t_i) \frac{W}{Q+R}$. Expanding $q(t_i)$ according to (1) results in

$$S(t_i) = \left((S(t) - i) \frac{Q}{W} + (i - 1) \frac{R}{W} - 2L \right) \frac{W}{Q + R}. \quad (3)$$

The dependency on i , the position of the child in the list of children of node t , can be eliminated by assuming $R \approx Q$. This is important because it will make it possible to solve the resulting recurrence relation. The assumption is realistic for many common situations like queries consisting of a set of keywords and results containing a URL. With this assumption, (3) can be simplified into the recurrence relation

$$S(t_i) = \left((S(t) - 1) \frac{Q}{W} - 2L \right) \frac{W}{2Q}. \quad (4)$$

Thus, setting a value for $S(t_r)$ for the root node t_r of the trader tree will transitively define the values for all other $S(t_i)$, where the resulting tree will have identical numbers of children for all nodes on the same tree level.

Let $S(j)$ identify the number of children of each node at level j , $q(j)$ the time a trader at level j has to answer a query. In other words, if a t_i is on level j , then $S(t_i) = S(j)$.

$S(j)$ can now recursively be computed based on the recurrence relation (4):

$$\begin{aligned} S(j+1) &= \left((S(j) - 1) \frac{Q}{W} - 2L \right) \frac{W}{2Q} \\ &= \frac{1}{2} S(j) - \frac{1}{2} - \frac{LW}{Q}. \end{aligned}$$

This recurrence is solved by the function

$$S(j) := 2^{-j} S(t_r) - 2 \left(\frac{1}{2} + \frac{LW}{Q} \right) (1 - 2^{-j}). \quad (5)$$

This function has to be seeded with $S(t_r)$, identifying the number of children the root trader has, which is determined by the maximum query execution time the root trader accepts according to (2), and the rest will follow.

3.2 Computing the Tree's Depth

Now the condition for a node that cannot have any children due to lack of time to forward queries will be formalized. This condition will be expressed as a threshold value for $S(j)$.

Then it will be proven that $S(j)$, and with it $q(j)$, are monotonically decreasing in j , both with a negative limit. This implies that for a positive number of children $S(t_r)$ for the root node t_r , there must be a first level j_{\max} on which traders are not given enough time to forward the query to any children and thus have to be searchables with no children instead of traders. The level j_{\max} will be computed based on the input parameters W (bandwidth), L (latency), Q (query size), and $S(t_r)$, the number of children the root node t_r has.

Once the tree depth can be computed, this result can be used to compute the total number of leaf nodes the tree has and the total number of network messages that the execution of one query causes.

A trader t at level j cannot have any children if the time it has to answer the query ($q(j)$) does not permit to forward the query to another node and receive a result, including network latencies for the sending and the receiving direction. Formally, this can be put as

$$q(j) < \frac{2Q}{W} + 2L \quad (6)$$

where again the assumption $R \approx Q$ is made. This enables the use of (5) in solving for j_{\max} . Substituting $q(j)$ according to (1) expresses this constraint in terms of $S(j-1)$, the number of t 's parent's children:

$$(S(j-1) - 1) \frac{Q}{W} - 2L < \frac{2Q}{W} + 2L$$

which can be solved for $S(j-1)$:

$$S(j-1) < 3 + \frac{4LW}{Q}. \quad (7)$$

When this condition holds, then the trader at level j cannot forward queries to any children, implying $S(j) = 0$ and defining j as the level of the leaves.

L , W , and Q can assume only positive values. With this, $S(j)$ is monotonically decreasing in j for

all $S(j) \geq 0$:

$$\begin{aligned} S(j) &= \left((S(j-1) - 1) \frac{Q}{W} - 2L \right) \frac{W}{2Q} \\ &= \frac{S(j-1) - 1}{2} - \frac{WL}{Q} \\ &= \underbrace{\frac{1}{2} S(j-1)}_{< S(j-1)} - \underbrace{\left(\frac{1}{2} + \frac{WL}{Q} \right)}_{> 0} \\ &< S(j-1) \end{aligned}$$

Therefore, as $q(j)$ from (9) is monotonous in $S(j)$, $q(j)$ is also monotonically decreasing in j .

Furthermore, $S(j)$ converges:

$$\lim_{j \rightarrow \infty} S(j) = -1 - \frac{2LW}{Q}. \quad (8)$$

And so does $q(j)$, the time a trader at level j has for computing its result. According to (1) (again assuming $R \approx Q$):

$$q(j) = (S(j-1) - 1) \frac{Q}{W} - 2L \quad (9)$$

and thus together with (8):

$$\lim_{j \rightarrow \infty} q(j) = \left(-2 - \frac{2LW}{Q} \right) \frac{Q}{W} - 2L. \quad (10)$$

As L , W , and Q can assume only positive values, this results in a negative limit for $S(j)$ and for $q(j)$.

Due to the monotony of $q(j)$ there is an index j_{\max} such that for all $0 \leq j < j_{\max} : q(j) \geq \frac{2Q}{W} + 2L$ and for all $j \geq j_{\max} : q(j) < \frac{2Q}{W} + 2L$. This j_{\max} is, as defined by (6), the level of the leaves.

j_{\max} can be computed from (5), (9), and (6) as follows. From (9) and (6):

$$q(j_{\max}) = (S(j_{\max} - 1) - 1) \frac{Q}{W} - 2L = \frac{2Q}{W} + 2L.$$

Substituting $S(j_{\max} - 1)$ according to (5) yields

$$2^{-j_{\max}+1} \left(S(t_r) + 2 \left(\frac{1}{2} + \frac{LW}{Q} \right) \right) \frac{Q}{W} = 6L + \frac{4Q}{W}.$$

Solving for j_{\max} brings

$$j_{\max} = 1 - \log_2 \frac{6LW + 4Q}{QS(t_r) + Q + 2LW}.$$

As j_{\max} can only assume integer values, the solution has to be "ceiled" as follows, identifying the first level that cannot forward to any children and thus contains the leaf nodes of the tree:

$$j_{\max} = \left\lceil 1 - \log_2 \frac{6LW + 4Q}{QS(t_r) + Q + 2LW} \right\rceil. \quad (11)$$

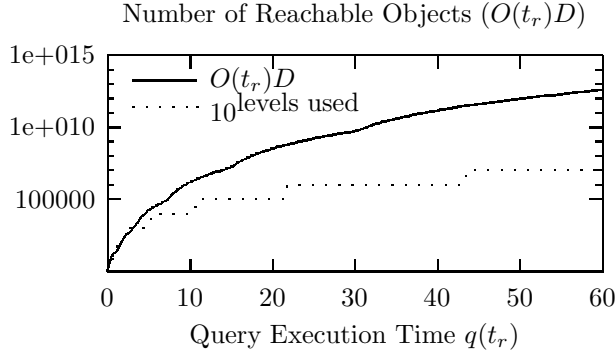


Figure 3: Number of searchable objects $O(t_r)$ to which a query can be forwarded given the maximum execution time $q(t_r)$ for the query.

The solid lines in figure 4 show a graph of $S(j)$ using the constraint that $S(j) = 0$ for $j \geq j_{\max}$.

Given the solution for j_{\max} the “cutoff point” for the product $O(t_r) = \prod_j S(j)$, telling the total number of leaf nodes included in the search by the root trader t_r , is known. The product can now be written as

$$O(t_r) = \prod_{j=0}^{j_{\max}-1} S(j). \quad (12)$$

Note the upper index $j_{\max} - 1$: Traders at level j_{\max} do not have any children, thus an $S(j) < 1$ and must hence not be included in the product. Figure 3 shows the results of using the solution for j_{\max} from (11) in (12).

The number of messages sent through the tree for a single request is $S(t_r) + S(t_r)S(1) + \dots + \prod_{j=0}^{j_{\max}-1} S(j) = \sum_{i=0}^{j_{\max}-1} \prod_{j=0}^i S(j) = S(t_r)(1 + S(1)(1 + S(2) \dots (1 + S(j_{\max} - 1)) \dots))$ which at the same time is the overhead as compared to the solution with $S(t_r) = O$, $j_{\max} = 0$ that minimizes the total number of messages sent per query (the centralized model). This can be seen as the *investment* into an increased value by either increasing the number of reachable objects or by increasing the currency of results, as will be shown later, or both.

3.3 Considering Query Frequency F

Up to now it has been assumed that a trader t can spend all its “network time” working on one query and consume up to the given maximum query execution time $q(t)$ for determining the corresponding search results. But this is not a very realistic assumption. If the time between queries $\frac{1}{F}$ is exceeded by the maximum query execution time $q(t)$, then the next query has already arrived before all subresults required for answering the previous query have been collected. But as it was assumed up to now that answering a query consumes all available network bandwidth, queries would have to get queued with the queue growing infinitely.

Therefore, an additional constraint has to be introduced. Each trader must not use more network time to answer a query than the time between arriving queries. Let $n(t)$ identify the *network time* that node t uses to answer a query. This is the time the network connection is used to forward the query to the $S(t)$ children and receiving as many responses. In the model as discussed so far the equation $n(t) = q(t)$ applied. The value for $n(t)$ computes as:

$$n(t) := S(t) \frac{Q + R}{W} \quad (13)$$

With this definition the constraint can be formalized as $n(t) \leq \frac{1}{F}$, or rewritten as a constraint on $S(t)$:

$$S(t) \leq \frac{W}{F(Q + R)}.$$

Note, that $n(t) \leq q(t)$, and that a trader can interleave processing of several queries to make optimal use of the available bandwidth.

Now how does this affect the shape of the trader tree? Based on the model described in section 2, at each trader two cases have to be distinguished:

- The trader is at a level j where by definition of the function $q(j)$ from (9) there is less time than $\frac{1}{F}$ to answer a query. This case is not critical and is handled correctly by the previous calculations.
- The previous function definitions for $q(j)$ from (9) allows the trader to take longer than $\frac{1}{F}$ to answer an incoming query. This case violates the additional constraint, and $S(t)$ has to be restricted to $\frac{W}{F(Q+R)}$.

It has been proven that $q(j)$, the time available for a trader at level j , decreases monotonically for growing j . This given, there is a first level j_F from which on $q(j) \leq \frac{1}{F}$ with $j \geq j_F$ and $q(j) > \frac{1}{F}$ for $j < j_F$. Trader nodes above this level have to constrain the number of children such that the maximum network time $n(t)$ for forwarding the query and receiving the results from these children is $\frac{1}{F}$:

$$S(j) = \frac{W}{F(Q + R)} \text{ for } 0 \leq j < j_F \quad (14)$$

Note, that the *absolute* time $q(j)$ these nodes have to answer the query is not affected by these considerations, and so is the recursive original solution for $S(j)$ from (5). All nodes below and including level j_F can exploit all of the available absolute time for network operations related to forwarding the query to children and receiving the corresponding results. Thus, for these nodes the regular function $S(j)$ as defined by (5) applies.

Number of Traders' Children $S(j)$ with and without the F Constraint

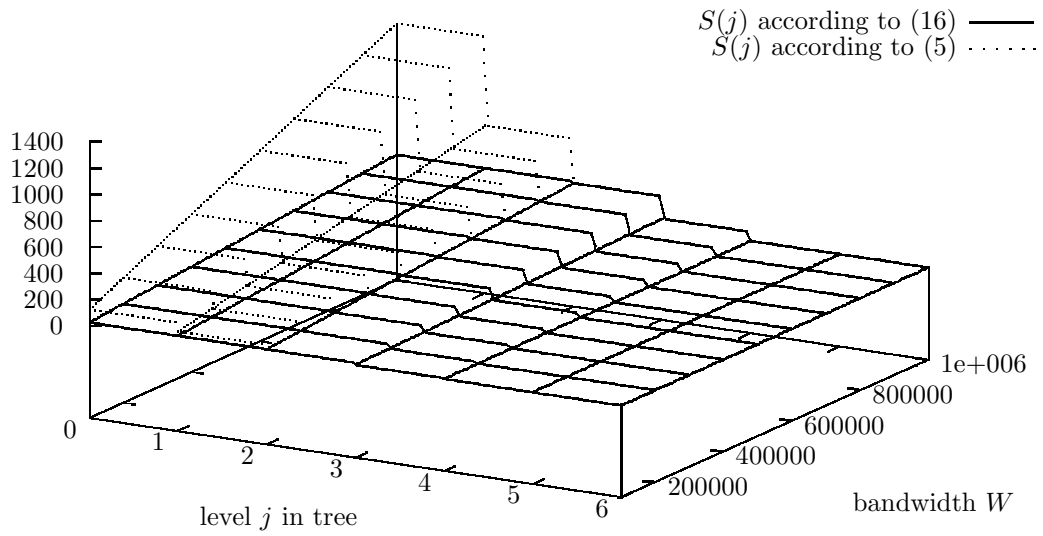


Figure 4: Traders' number of children with and without the additional constraint $q(j) \leq \frac{1}{F}$. Other assumed values: $L = 100\text{ms}$, $Q = 2000\text{bytes}$, $F = 1/s$, maximum allowed query execution time 5s.

Number of Reachable Searchables $O(t_r)$

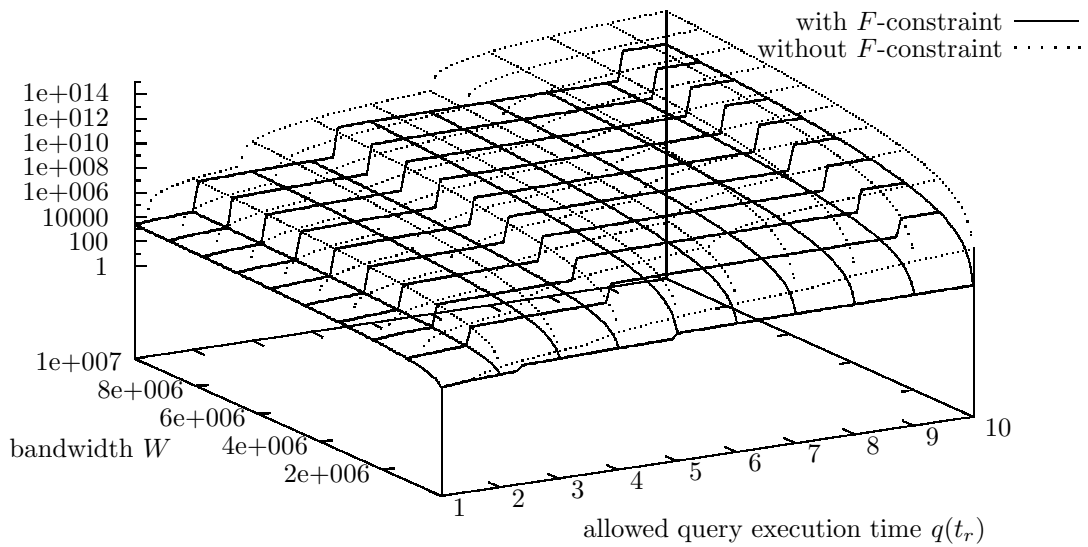


Figure 5: Number of reachable searchable objects (leaves in the trader tree) with and without the additional constraint $q(j) \leq \frac{1}{F}$. Other assumed values: $L = 100\text{ms}$, $Q = 2000\text{bytes}$, $F = 10/s$.

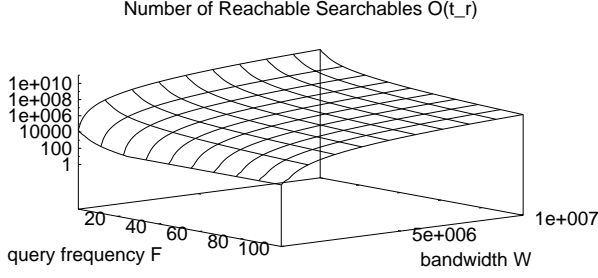


Figure 6: Number of reachable searchable objects (leaves in the trader tree) with the additional constraint $q(j) \leq \frac{1}{F}$ and plotted over the available bandwidth and the query frequency (in queries per second). Other assumed values: $L = 100\text{ms}$, $Q = 2000\text{bytes}$, $F = 10/s$.

The level number j_F can be computed using a similar approach as for j_{\max} before, again assuming $R \approx Q$, only instead of searching for the j that lets $q(j)$ become $\frac{2Q}{W} + 2L$ now the j_F is searched that lets $q(j)$ get smaller than $\frac{1}{F}$:

$$q(j_F) = (S(j_F - 1) - 1) \frac{Q}{W} - 2L = \frac{1}{F}.$$

Performing the same substitution as for j_{\max} before yields:

$$2^{-j_F+1} \left(S(t_r) + 2 \left(\frac{1}{2} + \frac{LW}{Q} \right) \right) \frac{Q}{W} = \frac{1}{F} + 4L + \frac{2Q}{W}.$$

Solving for j_F brings

$$j_F = 1 - \log_2 \frac{\frac{W}{F} + 4LW + 2Q}{QS(t_r) + Q + 2LW}. \quad (15)$$

The rewritten $S(j)$ definition considering this case distinction looks as follows:

$$S(j) = \begin{cases} j < j_F : \frac{W}{F(Q+R)} \\ j \geq j_F : 2^{-j} S(t_r) - \left(1 + \frac{2LW}{Q} \right) (1 - 2^{-j}) \end{cases} \quad (16)$$

where the value for j_F can be substituted according to (15). Figure 4 shows the difference between (16) and (5), figure 5 illustrates the impact on the total number of reachable leaves in the trader tree. As expected, only levels with smaller index (closer to the root) are affected, in this example levels 0 and 1. Section 3.4 will point at ways how a replication strategy for trader nodes close to the tree root can mend this situation.

3.4 Introducing Load Balancing and Bandwidth Increase

When a trader node has to constrain the time it uses the network for answering a query below the time that it has until it has to send out a result for the query ($q(t) < \frac{1}{F}$) then there are two typical approaches how this “bottleneck” can be alleviated:

- The trader’s network connection can be equipped with more bandwidth. This may be costly, though, and prices for additional bandwidth may rise even more than proportional.
- The trader may be split (*replicated*) into two or more with the tree connections being taken over by the replicas. Parent nodes will forward a query to one of the replicas using a round-robin schedule. In this case the traders resulting from the split may be positioned at completely separate and remote parts of the underlying network, such that affording the additional bandwidth does not suffer from the superlinear pricing for bandwidth when requested for one network location.

Depending mainly on the network latency there are differences between the two sketched approaches regarding the extra bandwidth. Assumed, the traders at level j have the following “query overload” x :

$$q(j) = x \frac{1}{F} \text{ with } x > 1.$$

Substituting $q(j)$ from (9):

$$(S(j - 1) - 1) \frac{Q}{W} - 2L = \frac{x}{F}. \quad (17)$$

Then if the traders at level j are overloaded by a factor x , by what factor b would their bandwidth W have to be increased in order to allow full use of the available time $q(j)$ for network operations again? The constraint then transforms into

$$(S(j - 1) - 1) \frac{Q}{bW} - 2L = \frac{1}{F}.$$

Multiplying both sides with x yields:

$$\left((S(j - 1) - 1) \frac{Q}{bW} - 2L \right) x = \frac{x}{F}.$$

Together with (17) this can be transformed into

$$b = x + \frac{2L(x - x^2)}{(S(j - 1) - 1) \frac{Q}{W} + 2L(x - 1)}. \quad (18)$$

If $L = 0$, then (18) degenerates to $b = x$. Furthermore, because of $x > 1$ and thus $x^2 > x$ and $2Lx > 2L$, and furthermore $S(j - 1) \geq 1$ for all $j < j_{\max}$ we have

$b < x$ for $L > 0$ because the fraction on the right-hand side of (18) is always negative.

Thus, in order to resolve an x -fold overload of a trader node, less than the x -fold bandwidth has to be invested when leaving the trader in place *without* splitting it into multiple nodes.

On the other hand, when the trader is split into x separate nodes, each of them requires bandwidth W , leading to a total bandwidth requirement of xW . When a trader node is to resolve the overload condition, then the pricing structure for bandwidth will decide whether to split or to afford more local bandwidth.

4 Updating Forward Knowledge

Up to now only the communication model down to the leaves has been discussed. This section will shed light on the bandwidth consumption and its implications of maintaining forward knowledge like inverse keyword indices. It will be shown how the required bandwidth for maintaining a searchable s 's forward knowledge depends on the permissible lack-of-currency measure $Y(s)$, the change frequency $C(s)$ of the underlying data, the size $A(s)$ of a message for updating the forward knowledge for one changed retrievable object, and the total number of retrievable objects $D(s)$ covered by searchable s (also see again table 1).

The resulting bandwidth can then be amortized over the query frequency F and can then be compared with the bandwidth required for forwarding a query. This enables quantifying the tradeoff between maintaining forward knowledge and forwarding queries to the source.

For simplicity it is assumed that a query can be answered based on the forward knowledge, without having to communicate further with the source on which the forward knowledge is based. Future extensions to this model may well consider forward knowledge types that cannot fully answer a query but may help in making good query routing decisions which also have a significant impact on bandwidth consumption.

Updates for a forward knowledge instance have to be transmitted from the source to the trader tree leaf s where the forward knowledge is maintained³. This has to happen every $\frac{D(s)}{Y(s)+C(s)-Y(s) \bmod C(s)}$ seconds on average in order to keep the index as current as demanded (see also figure 7). In this context, $a \bmod b$ is short for $a - b \lfloor \frac{a}{b} \rfloor$.

For simplicity it is assumed that updates are transmitted for each changed object. The average size for this update is $A(s)$. This results in an index update bandwidth of

$$\frac{A(s)D(s)}{Y(s) + C(s) - Y(s) \bmod C(s)}. \quad (19)$$

³This communication takes place "outside" the tree, as the sources are not an integral part of the trader tree.

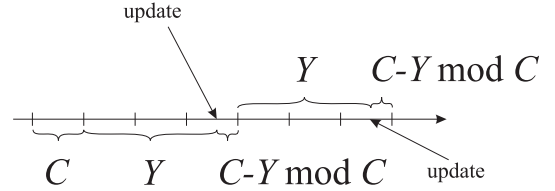


Figure 7: Index update frequency is based on the permissible lack of currency $Y(s)$ and time between changes $C(s)$. The parameter s has been omitted in the figure for brevity.

When the bandwidth for this gets larger than $2FQ$ then it would be more bandwidth-efficient to forward queries to the source instead of constructing and maintaining remote forward knowledge about the source. There will, though, be a tradeoff regarding the query execution time which will rise when the query has to get forwarded to the source.

Solving for F yields the query frequency threshold below which forwarding the query saves bandwidth as compared to maintaining forward knowledge for the source:

$$F < \frac{A(s)D(s)}{2Q(Y(s) + C(s) - Y(s) \bmod C(s))}.$$

Two general observations are confirmed and formalized by the above relation. Using forward knowledge stops saving bandwidth if the number of objects of the source to be represented in the forward knowledge grows too far, as the right hand side of the relation grows with $D(s)$; and using forward knowledge saves more bandwidth the higher the query frequency is.

5 Comparison with Central Index Approach

The model defined in section 2 covers the central index based approach as being implemented by today's typical Internet search engines. This approach is characterized in the model by a query execution time of 0s and thus a root "trader" with no children. The root in this case actually is a searchable and not a trader.

The constant $Y(s)$ in this case describes the permissible lack of currency of the search engine contents, $C(s)$ the average time between changes of a document indexed by the engine. F represents the number of queries users direct at the search engine. $D(s)$ is the number of documents the search engine has in its index. $A(s)$ is the average size of a document as indexed by the engine, because the whole document will have to get transferred in order to update the index appropriately.

Then the model provides the index update bandwidth under the assumption that the search engine does not have to visit documents that then will turn out not to have changed, which does not reflect the

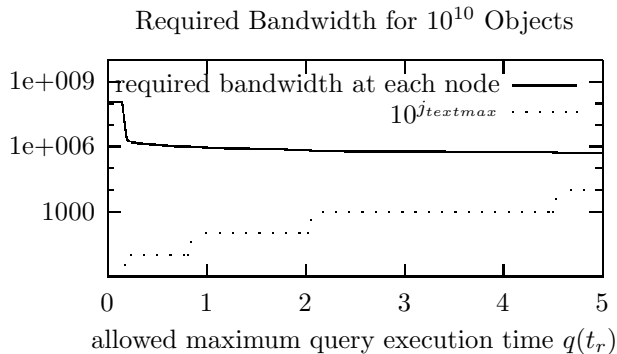


Figure 8: Required bandwidth per node for reaching 10^{10} objects with a lack of currency of at most one hour. Other assumed values: $L = 100\text{ms}$, $C = 10\text{days}$, $A = 10000\text{bytes}$, $Q = 2000\text{bytes}$.

way today's search engines proceed (see also subsection 5.1) and thus will provide a lower bound for the required bandwidth.

Figure 8 shows this extreme model on the far left where the allowed maximum query execution time is 0. The number of objects to be reached ($O(t_r)D$) is left constant at 10^{10} in this plot. It turns out that as soon as a second level is permitted by increasing the query execution time to roughly $\frac{1}{4}s$ the bandwidth required at each node goes down about two orders of magnitude and approximately remains like that even if the allowed query execution time is further increased.

This means that if the local bandwidth at the search engine is the limiting factor for its currency and coverage, then adding one or more levels and using the distributed tree model from section 2 can alleviate this bottleneck.

This suggests another question: When the bandwidth over all nodes participating in answering a query is accumulated, then how much accumulated bandwidth does a query consume per object reached? In other words this means, if bandwidth is associated with cost and the number of reachable objects with value then how must an instance of the model get customized in order to optimize the cost/value ratio.

Figure 9 shows this cost per object reached by the search ($O(t_r)D$ retrievable objects). The graph clearly suggests a modest (less than 1%) increase for the transition from only one level (central model) to two levels happening approximately at query execution time $.4s$. Using three levels imposes a slightly higher increase in required bandwidth per reached object of about 2.5%. Any further increase in the number of reachable objects does not significantly increase the bandwidth to be used per object.

Furthermore, figure 5 has visualized that a central model (in the figure indicated by query execution time < 1) can reach only several orders of magnitude fewer objects than a distributed model that, however, will

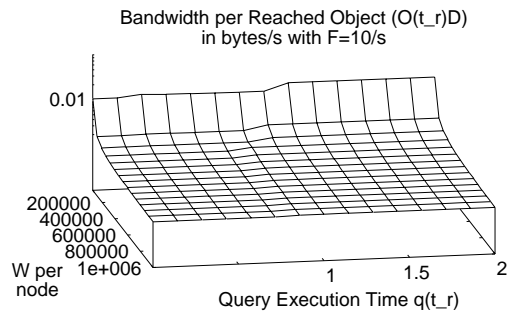


Figure 9: Required bandwidth per each of the $O(t_r)D$ objects reached; assumed values: $Y = 100\text{hours}$, $C = 10\text{days}$, $A = 10000\text{bytes}$, $Q = 2000\text{bytes}$, $F = \frac{10}{s}$.

require longer query execution times than those of the central approach. This holds even if the bandwidth for the central approach is increased beyond the limits of today's typical high-speed Internet backbones.

5.1 Polling vs. Updates on Change

A standard Internet search engine s does not know whether a document it has indexed has changed unless it loads the document again. Thus, in order to keep its index within the required maximum lack of currency time $Y(s)$ the engine has to load *all* documents it has indexed after the time $Y(s)$ has passed since the last update.

For simplicity, the conservative assumption may be made that the amount of data required for smart-updating an index for one document approximately equals the size of the document. According to table 1 this amount is termed $A(s)$. Further, let $D(s)$ be the number of documents the engine has indexed.

Then, the bandwidth required for keeping a polled index within the lack-of-currency interval $Y(s)$ is $\frac{A(s)D(s)}{Y(s)}$.

When an index is updated only when a document has really changed (which is assumed to happen every $C(s)$ seconds on average), according to (19) the required bandwidth is reduced to $\frac{A(s)D(s)}{Y(s)+C(s)-Y(s) \bmod C(s)}$. For small values for $C(s)$ — meaning high change rates — the bandwidth approaches that for polled indices. But as $C(s)$ grows, particularly beyond the value of $Y(s)$, the required bandwidth is asymptotically inversely dominated by $C(s)$.

In other words, if the average time between changes of a single document $C(s)$ is less than the allowed lack of currency $Y(s)$, then index updates that are performed only upon change save bandwidth. The savings can be quantified as

$$\frac{D(s)A(s)}{Y(s)} - \frac{A(s)D(s)}{Y(s) + C(s) - Y(s) \bmod C(s)}.$$

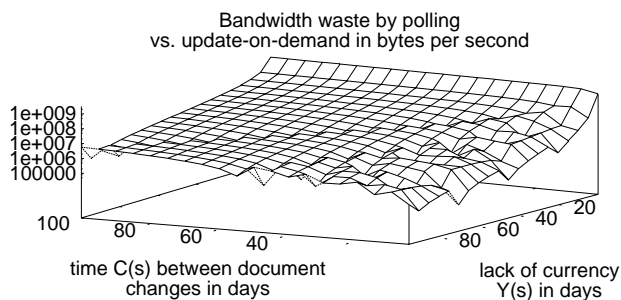


Figure 10: Bandwidth waste caused by polling the documents for maintaining the forward knowledge as opposed to getting notified only upon a change. Assumed values: $D = 10^{10}$ objects, $L = 100\text{ms}$, $A = 10000\text{bytes}$, $Q = 2000\text{bytes}$.

which for $C(s) > Y(s)$ can be simplified into

$$A(s)D(s) \frac{C(s) - Y(s)}{C(s)Y(s)}.$$

Figure 10 provides an overview of the impact of the savings (or, respectively, the waste). It becomes clear that for typical Internet-scenarios (e.g. $C(s) \approx 100\text{days}$, $Y(s) \approx 50\text{days}$) a search engine that tries to index 10^{10} objects will permanently waste bandwidth somewhere between 10^5 and 10^6 bytes per second.

6 Conclusions and Outlook

A formal model for quantitatively analyzing the domain of search from a bandwidth perspective in distributed information sources has been presented in this paper. The model assumes a possibly multi-rooted hierarchy of trader nodes with searchable information sources at the leaves of the tree. The provided formalism applies to a broad range of constellations and helps in finding the right number of subtraders in the trader tree and the maximum tree depth, making the indexing vs. query forwarding decision, and computing the tradeoff between buying more bandwidth and splitting a trader. Furthermore, the overhead of using polling instead of using a change-based notification mechanism has been quantified. The model uses a multi-dimensional parameter space including the query frequency, query and result object sizes, network bandwidth and latency, change rate of the searchable information, and the requested currency of search results.

The model formally confirms that a centralized approach does not scale. Deeper hierarchies do not significantly increase the bandwidth required per covered object, but they allow numbers of objects that are several orders of magnitude larger than those reachable with a centralized approach. The tradeoff is a slightly increased query execution time. It has to be emphasized, that the key result is the *formalism*, not necessarily some of the obvious statements that can be retrieved when applying the formalism to a typical set of parameters.

With the presented model it is now possible to set up a cost model for Internet search, shaping the “economy of search” in the Internet. From user and trader preferences a total cost or value can be assigned to a single query up-front. Eventually, this will help to decide whether a search service can be offered for free or the service provider has to charge a fee, and it helps in defining search architectures that meet a given business and cost model.

An important issue for further research is to compute the optimal number of result objects a searchable source should return, depending on the model parameters and the total number of desired result object, and how this number in turn affects the optimal tree shape.

References

- [1] Steve Lawrence and C. Lee Giles, “Searching the World Wide Web,” *Science*, vol. 280, no. 5360, pp. 98, 1998.
- [2] Anthony S. Tomasic, “Distributed queries and incremental updates in information retrieval systems (thesis),” Technical Report TR-458-94, Princeton University, Computer Science Department, June 1994.
- [3] D. Culler, R. Karp, D. Patterson, A. Sahay, K. Schauer, E. Santos, R. Subramonian, and T. von Eicken, “LogP: towards a realistic model of parallel computation,” Computer science division report, University of California, Berkeley, 1992.
- [4] Paraskevi Fragopoulou and Selim G. Akl, “Optimal communication algorithms on star graphs using spanning tree constructions,” *Journal of Parallel and Distributed Computing*, vol. 1, no. 24, pp. 55–71, 1995.
- [5] Leqiang Bai, Hajime Maeda, Hiroyuki Ebara, and Hideo Nakano, “A broadcasting algorithm with time and message optimum on arrangement graphs,” *Journal of Graph Algorithms and Applications*, vol. 2, no. 2, pp. 1–17, 1998.
- [6] Axel Uhl, “The future of Internet search,” in *DOA’01 International Symposium on Distributed Objects and Applications, Short Papers*, Roberto Baldoni, Ed., Sept. 2001.
- [7] Axel Uhl and Horst Lichter, “New Wave Searchables: Changing the paradigm of Internet scale search,” in *International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet*, L’Aquila, Italy, Aug. 2001, SSGRR.
- [8] C. Weider, J. Fullton, and S. Spero, “RFC 1913: Architecture of the whois++ index service,” Feb. 1996.