Distributed Information Discovery in Networks from the Perspective of Information Theory

Hong Huang
Klipsch School of Electrical and Computer Engineering
New Mexico State University
Las Cruces, NM88003
hhuang@nmsu.edu

ABSTRACT

In distributed information discovery, object information is discovered through a number of steps, looking up a routing table in each step, as in peer-to-peer (P2P) systems. This paper aims to establish a connection between information theory and distributed information discovery. Such a connection exists inherently and can be applied fruitfully, but has so far attracted not much attention in the research community. In the paper, we present some preliminary results in establishing this connection and hope to attract further investigation. We have made three contributions. First, using information theoretical arguments, we establish fundamental lower bounds on table sizes and step numbers. These bounds go beyond those derived previously and can deal with object popularity. Second, we provide some insight on the relationship between achieving lower bounds and a closely related issue, i.e., balancing load, and, in passing, point out the equivalence between tree-based and ring-based P2P systems. Third, we establish the analogy between search sequences and data compression and coding, and propose a distributed implementation of Shannon code that can reduce the expected length of search sequences to an arbitrarily small value (depending on the object popularity) at the cost of at most doubling the table sizes.

1. INTRODUCTION

An important function of a network is to discover information about an object such as a route, an address, a piece of distributed data, etc. There are a variety of ways to discover object information in a network. At one extreme, object information is distributed to every node in the network, as in link state routing in traditional networks. At the other extreme, object information is not distributed and needs to be searched, as in on-demand routing in mobile ad hoc networks. The two extremes have scalability limitations, because a complete distribution of information at one extreme or a blind search at the other incurs a cost on the order of that of a broadcast. More promising in scalability is the approach between the two extremes that relies in part on distribution of information and in part on search, as in information lookup schemes in peer-to-peer (P2P) networks [2][3][4][5]. We call such an approach distributed information discovery since objection information is distributed inside the network and is discovered through a search sequence. We focus on distributed information discovery in this paper.

Given object information is distributed in the network, it is sensible to ask how to measure, quantify, and analyze such information, and how the distributed information is related to the uncertainty (entropy) of the object. Information theory provides a fitting and powerful framework to reason about such questions.

The goal of this paper is to establish a connection between information theory and distributed information discovery in networks. Such a connection exists inherently and can be applied fruitfully, but has so far attracted not much attention in the research community. This paper presents some preliminary results in establishing this connection and hopes to attract further investigation. A related, but different, previous work is about using information theory to analyze routing protocol overhead [6].

We made three contributions in this paper, described in Sections 2, 3, 4, respectively. First, using information theoretical arguments, we establish fundamental lower bounds on table sizes and step numbers. These bounds go beyond those derived previously and can deal with object popularity. Second, we provide some insight on the relationship between achieving lower bounds and a closely related issue, i.e., balancing load, and, in passing, point out the equivalence between tree-based and ring-based P2P systems. Third, we establish the analogy between search sequences and data compression and coding, and propose a distributed implementation of Shannon code that can reduce the expected length of search sequences to an arbitrarily small value (depending on the object popularity) at the cost of at most doubling the table sizes.

Some basic concepts and definitions in information theory are included in the Appendix, and a good reference book can be found in [1].

2. AN INFORMATION THEORETICAL FORMULATION AND BOUNDS ON TABLE SIZES AND STEP NUMBERS

We consider the problem of search for an object in a network of n nodes. The location of the object is represented by a random variable x taking values in the alphabet $A = [1, 2, \dots n]$. Before the search, x can be in any of the n nodes, and this uncertainty is measured by the entropy $H(x)$, called *object entropy*. The search is successful when this entropy is removed. $H(x)$ assumes the maximum value of log$n$ (base 2 is implied if not explicitly stated), when the object assumes uniform distribution (so-called maximum entropy distribution) [1].

Information about the object is stored in tables in a distributed manner inside the network. The outcome of looking up table i is represented as a random variable y_i. The entropy of y_i, $H(y_i)$, called *lookup entropy*, is removed after the look-up. For a table with m_i possible outcomes, i.e., a table with m_i rows, the maximum value of $H(y_i)$ is logm_i, which occurs when the outcome assumes uniform distribution.

We model a search as a sequence of table lookups. Specifically, the search is carried out by a k-step sequence, represented by the corresponding outcomes of table lookups $[y_1, y_2, \ldots y_k]$. The search is successful when the cumulative entropy reduction by the table lookups equals $H(x)$, the object entropy.

Because of redundancy of information among tables, the entropy reduction at the ith step, ΔH_i, is not equal to the lookup entropy $H(y_i)$, but needs to be computed from definition, taking into account information acquired in the previous history up to (i-1)th step. Thus, using the notation $y^i = [y_1, y_2, \ldots y_i]$, we have,

$$\Delta H_i = H(x \mid y^{i-1}) - H(x \mid y^i) = I(x, y_i \mid y^{i-1}) \tag{1}$$

In the above, $H(x|y^i)$ is the conditional entropy of x, given y^i; and $I(x,y_i|y^{i-1})$ is the mutual information between x and y_i, given y^{i-1}, which provides another definition of ΔH_i.

Adding entropy reductions along the search sequence, making use of (1), we have,

$$\sum_{i=1}^{k} \Delta H_i = H(x) - H(x \mid y^k) = H(x) \tag{2}$$

The last equality holds because the search succeeds at the kth step and x is determined, i.e., $H(x|y^k) = 0$. Equation (2) essentially provides the condition for a search to succeed in terms of sum of entropy reductions at steps. We provide a bound on ΔH_i in the following.

Lemma 2.1 *Upper bound on ΔH_i:*

$$\Delta H_i \le H(y_i) \le \log m_i \tag{3}$$

Proof: We expand $H(x, y_i|y^{i-1})$ in two different ways,

$$H(x, y_i \mid y^{i-1}) = H(x \mid y^{i-1}) + H(y_i \mid x, y^{i-1})$$
$$= H(y_i \mid y^{i-1}) + H(x \mid y^i)$$

Rearranging terms around the second equality, we obtain,

$$\Delta H_i = H(y_i \mid y^{i-1}) - H(y_i \mid x, y^{i-1})$$
$$\le H(y_i \mid y^{i-1}) \le H(y_i) \le \log m_i$$

The first inequality in the above comes from the fact that entropy is nonnegative; the second inequality from the fact that conditioning can not increase entropy; and the last inequality from the fact that entropy is maximized by a uniform distribution (thus the value of logm_i). *Q.E.D.*

Lemma 2.1 implies that the entropy reduction capacity of a table is maximized when the lookup outcomes are equally likely and there is no mutual information between the outcome of the current lookup and those of previous ones. Combing equations (2) and (3), we obtain the fundamental lower bounds on table sizes and number of steps as below.

Proposition 2.2 *The lookup table sizes and step numbers of a search sequence are lower-bounded by the following expression,*

$$\sum_{i=1}^{k} \log m_i \ge H(x) \tag{4}$$

Corollary 2.3 *If the lookup tables have a uniform size, m, then the number of steps, k, is lower-bounded by the following expression,*

$$k \ge \frac{H(x)}{\log m} \tag{5}$$

When the object distribution is uniform, which is the maximum entropy distribution, we have $H(x)$ = log$n$, and we rewrite (5) as,

$$k \log m \ge \log n \tag{6}$$

Then we can immediately draw the following conclusion:

Corollary 2.4 *If object distribution is uniformly random, then (a) to have constant table size, k must be O(logn); (b) to have constant step number, m must be $O(n^{1/k})$; (c) it is impossible to have both constant table size and step number.*

The conclusions similar to the above are already known to researchers in areas of hierarchical routing, compact routing, P2P networks, etc. Let us look at a few examples. Information look-up schemes in P2P networks roughly correspond to case (a) in Corollary 2.4 (*O(logn)* steps, table size of *O(logn)*) [2][3][4]. Compact routing corresponds to case (b) (2 steps, table sizes of $O(n^{1/2})$ and $O(n^{1/2}\log n)$) [9]. So is flat routing (1 step, table size of n).

Corollary 2.4 recovers lower bounds established previously by combinatorial and graph theoretical arguments [7][8]. However, our results are derived from simple information theoretical arguments, with minimum assumption about particular routing/lookup models, thus are more universal. More importantly, our result in Proposition 2.2 is expressed in terms of object entropy, which goes beyond the bounds established by previous work and can deal with object popularity. We will comeback to the last point in Section 4. The lower bounds in Corollary 2.4 can be achieved simply by a tree-based hierarchical scheme if load balance is not required, which is of course not adequate for a practical system. In the next section we analyze the relationship between balancing the load and achieving the lower bounds.

3. ACHIEVING LOWER BOUNDS AND BALANCING LOADS

We can consider a search sequence in a network of n nodes as a process that narrows down the cardinality of the *candidate set* (defined as the subset of nodes that the object belongs) from n to 1. So the search sequence can be represented as a candidate set sequence $[n_1, n_2, \dots n_k]$, where n_i is the cardinality of candidate set at step i and $n_0 = n$, $n_k = 1$. Assuming maximum entropy (uniform) distribution, the entropy reduction at step i can be written as: $\Delta H_i = \log n_{i-1} - \log n_i$. Therefore the bounds in Proposition 2.2 is achieved if and only if $\Delta H_i = \log m_i$, i.e., $m_i = n_{i-1}/n_i$. This can be easily identified as a hierarchical scheme that partition the candidate set m_i ways at level i, with number of nodes at level i being $n_i = n/m_1 m_2 \dots m_{i-1}$.

The problem with a traditional hierarchical scheme is, of course, imbalance of load. Here we provide some analysis. In the following we assume the table sizes are the same (m) for clarity, but the results can be easily extended to the case of uneven table sizes. Let v_i be the number of varieties of tables at the ith level, each variety corresponding a particular prefix of the table or the outcomes of previous i-1 lookups. At the first/root level, there is no prefix, so $v_1=1$. At the ith level, there are m^{i-1} varieties of tables, each corresponding to a particular i-1 prefix. So we have $v_i=m^{i-1}$. Since *a query visits exactly one variety of tables at each level*, we have a geometric load distribution across varieties: the load per variety at the ith level is: $r_i=r/m^{i-1}$, where r is the total query rate in the network. Note that *the load-imbalance problem pertains to low-level tables, not to particular nodes unless they coincide with each other*.

The method to achieve load-balance, not surprisingly, is replication, which is used in P2P lookup schemes. The idea is to replicate low-level tables with high load. In tree-based P2P schemes such as Pastry [4], a lookup table can be considered as a $k \times m$ matrix. The ith row of the matrix corresponds to a table at the ith level in a hierarchy. A table at each level is available at every node. The ith level tables, which have m^{i-1} varieties, are replicated c_i times, where $c_i=n/m^{i-1}=m^{k-i+1}$. Thus the number of tables at each level, t_i, is the same, since $t_i= v_i \times c_i=n$ (given $v_i=m^{i-1}$). Now the load at a table of a certain variety at the ith level becomes: $r_i=r/t_i=r/n$. In this way, *perfect load balance*[1], defined as load uniformly spread across all nodes, is achieved at the cost of inflating the table sizes at each node by a factor of m. A by-product of replication of tables is the flexibility to select neighbors and routes.

Another popular lookup scheme, Chord [2], which is called ring-based, can in fact be shown to be equivalent to a tree-based scheme with m=2. This is because the lookup table of a ring-based scheme can be obtained by a transformation of that of tree-based scheme, which is described in the following. Let us start with a tree-based scheme with its $k \times m$ lookup matrix. We first normalize the key to be looked up by subtracting it by the key representing the local node, which makes all the lookups descend along the last column of the matrix to an appropriate row (assuming the last column corresponds to all zeros, which is the key of the local node). This normalization can be easily undone after the lookup. Now we have a normalized lookup table as shown below, where the number inside a bracket at level i indicates n_i, the cardinality of the candidate set at that level:

$$\begin{array}{c}
[n/m][\ n/m]\ldots\ldots\ldots\ldots[\ n/m] \\
\hfill /\backslash \\
[n/m^2]\ [n/m^2]\ldots\ \ldots\ldots[n/m^2] \\
\hfill : \\
\hfill /\backslash \\
[n/m^k]\ [n/m^k]\ldots\ \ldots\ldots.[n/m^k]
\end{array}$$

The above matrix can be considered as a candidate set representation of a lookup process, with the cardinality of each level m times smaller than that of the previous level, and that of last level being 1, indicating the search succeeds. Let us consider a tree-based scheme with m=2, $k=\log n$, with a $k \times 2$ matrix normalized lookup table. We can stretch the matrix to a linear arrangement by embedding the row at level i into one of two partitions it belongs at level i-1, as shown below:

$$[n/2[\ n/2^2[n/2^4[\ldots\ldots\ldots.[\ n/2^{\log n}]]\ldots]$$

The above linear arrangement is nothing but the candidate set representation of Chord. Such embedding can be easily extended to arbitrary m values. To lookup a key, a tree-based scheme can use longest prefix match just like the greedy interval match in a ring-based scheme. Thus we have shown that *there is a linear embedding of a matrix that transforms lookup tables from a tree-based scheme to a ring-based scheme, i.e., lookup tables of the two schemes provide the same information. In that sense the tree-based and ring based schemes are equivalent.*

An optimization on the above schemes is a P2P lookup scheme based on de Bruijn graphs [8], which has the benefit of achieving load balance without inflating table sizes. The idea is to reuse the table at a single level for different levels at different steps. In such a scheme, each node maintains a table of size m. The table serves as level 1 table for a local query, and simultaneously serves as level i table for a query that visits it at the ith step. Such table has $v_i=m^{k-1}$ varieties at all levels, each corresponding to a particular k-1 character prefix of the node. Since there are m nodes with the same k-1 character prefix, we have $c_i=m$, $t_i= v_i \times c_i=n$, and $r_i=r/t_i=r/n$, again obtaining perfect load balance and simultaneously achieving the bound in Corollary 2.4 (because equation (6) is satisfied here).

4. INFORMATION LOOKUP AND DATA COMPRSESSION/CODING

In this section, we describe the connection between data compression and coding in information theory and information lookup and replication based on popularity. The motivation is to place information in such way that less number of steps is required for more popular objects. Basically we are faced with an optimization problem: how to

[1] It is in the average sense and does not include fluctuations in node ownership of ID space [11].

distribute information so that the expected number of steps to find an object is minimized.

We can view the outcomes of a successful search sequence $Y_j = [y_1, y_2, \ldots y_{kj}]$ of length k_j as a code for the location x_j of object j, since Y_j uniquely determines x_j. Let p_j be the probability (popularity) that a random request is for the object j, m be size of lookup table (assuming uniform table size), then the optimization problem can be stated as,

$$\min E[k] = \sum_j p_j k_j \quad \text{subject to} \sum_j m^{k_j} \le 1 \tag{7}$$

In the above, the constraint is the Kraft's inequality, which applies to any uniquely decodable code [1]. The optimization problem is the classical minimum description length (MDL) problem in source coding theory [1]. The solution to the optimization problem is the famous Shannon code, with code length of object j given by,

$$k_j = \left\lceil \log_m \frac{1}{p_j} \right\rceil \tag{8}$$

In the above, a round-off operation is taken because the step number is an integer. Writing $H_m(x)$ as the entropy of x with log base m, the expected code length satisfies,

$$E[k] < H_m(x) + 1 \tag{9}$$

A simple, approximate method to implement such a Shannon code in a distributed manner in a network is to replicate information about an object at the step or code length prescribed by equation (8), i.e., as a log function of object popularity. The question is how to obtain object popularity. Methods have been proposed to estimate popularity of objects in a distributed manner, for example, by aggregating and disseminating object access frequencies or inter-arrival times [10]. Here we provide a simple protocol whose advantages will be shown later.

Our protocol is based on P2P lookup schemes that have a matrix presentation of lookup tables, which covers tree-based, ring-based, de-Bruijn-graph-based schemes as shown in the previous section. Recall that in such a P2P lookup scheme, a routing table at a node is arranged in a $k \times m$ matrix, whose ith row responds to a lookup at the ith step (in the case of de-Bruijn-graph-based scheme, the same row is reused). Our protocol to approximately implement a Shannon code in a network is summarized below.

Protocol 4.1 *Access frequency $f_{j,i}$ of object j at the ith step of the search is maintained during a time window. If i is the smallest step number such that $f_{j,i} \ge 1/m$ holds, then, the information about object j is replicated (if it is not already present) at this node at the ith row of the matrix.*

Justification: We first estimate the popularity $p'_{j,i}$ of object j with respect to a node that it visits at the ith step, then estimate the corresponding step number $k'_{j,i}$ at which the object is to be replicated. Here we refer the popularity to both the object and the step (table), because there is an issue of correctly estimating popularity, which we will discuss later. Now, according to the principle of conditional probability we have,

$$p'_{j,i} = f_{j,i} p'(y^{i-1})$$

In the above, $f_{j,i}$ is an estimate of the probability of querying object j conditioned on the fact the object has a prefix of y^{i-1}. The probability of object having a prefix of y^{i-1}, $p'(y^{i-1})$, can be estimated as, assuming equally likely prefixes, $p'(y^{i-1}) = 1/m^{i-1}$. Given $f_{j,i} \ge 1/m$, so we have,

$$p'_{j,i} = f_{j,i} p'(y^{i-1}) \ge \frac{1}{m} \cdot \frac{1}{m^{i-1}} = \frac{1}{m^i}$$

$$k'_{j,i} = \left\lceil \log_m \frac{1}{p'_{j,i}} \right\rceil \le \left\lceil \log_m \frac{1}{m^i} \right\rceil = i$$

According to the protocol, the information about object j is replicated at the smallest i at which the condition $f_{j,i} \ge 1/m$ is satisfied. So Protocol 4.1 guarantees that the number of steps to locate the object is no larger than that prescribed by equation (8). Also, the fact that replication occurs at the smallest i that satisfies the replication condition prevents the query continuing downstream to visit nodes with larger i values (number of steps). Therefore there is no double counting of visit frequencies.

Protocol 4.1 has three advantages First, the protocol is very simple, and requires minimal change to the existing P2P lookup schemes. Second, object popularity is estimated strictly locally with no requirement for message exchange, saving a great deal of communications bandwidth. Third, perhaps most importantly, the protocol provides *the correct way to estimate popularity of objects* with respect to tables. *There are two kinds of popularity: global (perceived by the network) and local (perceived by the individual nodes); and they are obviously not equivalent*. Some objects are popular to certain nodes but not to others. The correct way to estimate the popularity of an object, which is used by Protocol 4.1, should be based on the local perception. As an example, an object can be globally unpopular, but is queried more than $1/m$ percentage of time at a particular node with esoteric taste. According to Protocol 4.1, the information of this object is replicated at the local level, but would not be replicated if global popularity is the criterion.

The cost of Protocol 4.1 is at most doubling the table sizes. The reason is that at most m objects with visit frequency of at least $1/m$ can be replicated at each row of size m.

The benefit of Protocol 4.1 is a significant reduction in expected number of steps to locate an object. In addition, the number of steps in the worst case is the same as that of the base P2P scheme. This is different from a standard Shannon code where the worst case code length (for unpopular objects) can be very large. Quantitatively, suppose we have m_0 objects with popularity of at least $1/m$, m_1 objects with popularity of at least $1/m^2$, etc. The expected number of steps $E[k]$ can be computed as,

$$E[k] \le \sum_{i=1}^{k} i \frac{m_i}{m^i} \tag{10}$$

In the trivial case of *m* objects, each having a popularity of *1/m*, the expected number of steps is zero, since they are replicated at the local level. In other words, depending on the popularity distribution of objects, Protocol 4.1 can reduce the expected number of steps to an arbitrarily small value.

5. CONCLUSION AND FUTURE WORK

In this paper, we have shown the connection between information theory and distributed information discovery, and demonstrated some preliminary results profited from such a connection. We believe much more can be accomplished by further exploiting the connection. Our future work will be on the following line of questioning. Protocol 4.1 corresponds to a source code in information theory. One naturally would ask: In the context of distributed information discovery, what corresponds to a channel code? Do we have a separation theorem of source and channel codes? How about rate distortion theory? What is the connection with network coding? etc.

6. REFERENCES

[1] T. M. Cover and J. A. Thomas. *Elements of information theory*, 1st Ed. Wiley-Interscience, New York, 1991.

[2] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of ACM SIGCOMM'01*, San Diego, CA, 2001, pp. 149–160.

[3] B. Y. Zhao, J. Kubiatowicz, and A. Joseph. *Tapestry: An infrastructure for fault-tolerant wide-area location and routing*. Technical Report UCB/CSD-01-1141, U.C. Berkeley, 2001.

[4] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of IFIP/ACM International Conference on Distributed Systems Platforms*, Heidelberg, Germany, Nov. 2001, pp. 329–350.

[5] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of ACM SIGCOMM'01*, San Diego, CA, 2001, pp. 161–172.

[6] N. Zhou and A.A. Abouzeid. Routing in ad hoc networks: a theoretical framework with practical implications. In *Proc. of IEEE INFOCOM'05*, Miami, FL, 2005, March 13-17, pp. 1240-1251.

[7] Xu, J., Kumar, A., and Yu, X. On the fundamental tradeoffs between routing table size and network diameter in peer-to-peer networks. *IEEE Journal on Selected Areas in Communications*, vol 22, no 1, pp. 151--163, Jan 2004

[8] D. Loguinov, A. Kumar, V. Rai, and S. Ganesh. Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience. In *Proc. of ACM SIGCOMM'03*, Karlsruhe, Germany, August 2003.

[9] M. Thorup and U. Zwick. Compact routing schemes. In *Proc. of 13th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, ACM PRESS, July 2001

[10] V. Ramasubramanian and E. G. Sirer. Beehive: exploiting power law query distributions for *O(1)* lookup performance in peer-to-peer overlays. In *Proc. of Symposium on Networked Systems Design and Implementation*, San Francisco CA, 2004.

[11] B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp and I. Stoica, Load balancing in dynamic structured P2P systems. In *Proc. of IEEE INFOCOM'2004*, March, 2004.

APPENDIX

BASIC CONCEPTS IN INFORMATION THEORY

In information theory, the uncertainty about a discrete random variable *x* with alphabet *X* is measured by its entropy *H(x)*, defined as,

$$H(x) = -\sum_{x \in X} p(x) \log p(x)$$

The uncertainty about *x* given another random variable *y* with alphabet *Y* is measured by the conditional entropy *H(x|y),* defined as (*p(x,y)* and *p(x|y)* are joint and conditional probabilities, respectively),

$$H(x \mid y) = -\sum_{x \in X, y \in Y} p(x,y) \log p(x \mid y)$$

The uncertainty about two random variables *x* and *y* is measured by the joint entropy *H(x,y),* defined as,

$$H(x \mid y) = -\sum_{x \in X, y \in Y} p(x,y) \log p(x,y)$$

The reduction in uncertainty of *x* due to the knowledge of *y* is measured by the mutual information *I(x;y)*, defined as (with multiple equivalent definitions),

$$I(x;y) = H(x) - H(x \mid y) = H(y) - H(y \mid x)$$
$$= H(x) + H(y) - H(x,y)$$