ICN: Interest-based Clustering Network

Xiaole Bai, Shuping Liu, Peng Zhang, Raimo Kantola
Networking Laboratory
Helsinki University of Technology
{xbai, sliu1, pgzhang, kantola}@netlab.hut.fi

Abstract:

An Interest-based Clustering peer-to-peer Network (ICN) architecture is introduced in this paper. ICN uses a lot of Freenet mechanisms and is based on cache management. ICN is self-organizing, fully distributed, scalable, and logically hierarchical. In ICN, the upper level is bound by de Bruijn graph. Nodes in the lower level self-cluster based on interest. Through analysis and simulation, ICN shows good fault-tolerance, efficient data retrieval and resource usage as well as low overhead traffic.

1. Introduction

In recent years, many peer-to-peer (p2p) schemes ([1-9] etc.) have been proposed for file sharing or distributed computing. However, none of these proposals can support all of the following features probably required in a global system: 1) A distributed control mechanism. This follows from the requirement for scalability. Besides, only a distributed system can be highly fault-tolerant. 2) Efficient data retrieval. First, the less overhead traffic the better. Second, a quick response for newly published resources, a fast search time and overall stability of the data resource are desired. 3) A robust architecture and service. The architecture should not fail due to the failure of a small number of critical elements. Under adverse circumstances, for example, when under DDOS attack, service quality should not degrade sharply. 4) Efficient resource usage. Both network resources, like total network bandwidth, and local machine resources should be used efficiently.

Sharing the common interest of trying to provide a better solution for the above, a user-friendly p2p proposal is introduced in this paper. Particularly, the starting point for our proposal is human interests.

We note that in real world usually people do have distinctive preferences and taste. For example, among those who like music, some are interested in heavy metal, some like classical, while others like blues or jazz. A reading club survey result shows that, some students like storybooks while others like research books [10]. Furthermore, these storybooks can be classified into five small categories according to different student preferences. We further note that such taste does not change dramatically over time. If you have some preference, it may change, but usually the change is very slow. The survey [10] shows that the preference for love book is getting stronger when students are maturing. And [11] also shows that some interest, e.g. for computer technology, may change with people's age. Preference and its very slow change compared to the computing time scale are two starting points for our peer networking proposal.

In our Interest-based Clustering p2p Network (ICN), contents are classified and human interest is considered for clustering. ICN is more humanized, and it shows advantages on both functionality and as a distribution channel for commercial content.

The map of this paper is as follows. Next, we take a look at related work. In Section 3 we introduce the construction of ICN, and the way it works. In Section 4 an analysis and evaluation are given. Finally, we present some conclusions and our future work.

2. Related work

Gnutella[12] and Freenet[8] are two well-known, large scale, fully decentralized directory and distribution systems. Although, they both belong to the random DHT (Distributed Hash Table) class of architectures, they use different mechanisms to locate and retrieve shared files. Gnutella uses broadcast while Freenet uses chain routing. One advantage brought by the decentralized indexing network architecture is the inherent scalability. Additionally fault tolerance is improved. The main disadvantages are slow information discovery and added query traffic on the network.

Gnutella faces serious scaling and reliability problems when the network is very large [13]. Although Freenet is of better scalability, it provides a very limited sort of directory service putting search efficiency in question. Lookup process may traverse significant portions of the entire peer space. In Freenet, data replication along the retrieval path improves reliability and robustness. But such treatment without discrimination will result in low efficiency for local resource usage. A simple example will make this clearer: Node A is interested in movies. If its routing table and the cached content are mostly movie-related, it will definitely facilitate A's search. However, the fact that A is on the path of several nodes that are interested in pictures makes A have to keep a lot of information about pictures.

Chord[5], CAN[2], Oceanstore[3], etc., belong to the class of deterministic DHT architectures where the assignment of a key to the node is determined by the structure of the key space. They all have some graph theoretic background making search more efficient than what is achieved by random DHT schemes. However, their highly structured approach makes them vulnerable to malicious users.

ICN uses novel cache management, shares the common routing strategy used in Freenet, and is logically hierarchical. But by nature it still belongs to the class of random DHT architectures. De Bruijn graph is used to build a deterministic DHT relationship to facilitate content locating in the upper layers of ICN.

3. The architecture of ICN

Interest based Clustered Network (ICN) has three levels, as shown in Figure 1.

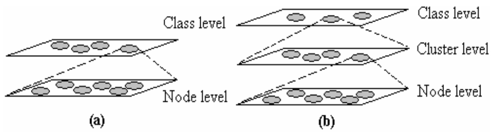


Figure 1. ICN may have two or three logical layers.

Descriptions in this section are based on three-layered ICN.

The top is the class level. Each class captures a big content category: music, movie, pictures and so forth. The middle level contains the clusters. Clusters, which are defined in advance, are small categories contained in classes. The node level is the lowest. To illustrate the use of levels, let's look at an example. A class can be *Music* in which there are many clusters like *Heavy Metal*, *Classical*, *Blues*, etc. In the *Classical* cluster, there are a large number of users who are interested in classical music and will publish and look for classical music resources.

Although the number of levels could be larger than three and as a result, we could pinpoint human interest more accurately, we prefer to give ICN quite a flat architecture with just three or two layers. We believe flatter architecture will reduce traffic for clustering and re-clustering especially when user volumes are very large.

A node can register in different clusters of classes at the same time.

A special class, *null-class*, is created for the new coming nodes and those that show no preference. This class directly contains nodes at the third level, that is, no clusters. In each class, except in the *null-class*, there is a special cluster, *null-taste* for those users, for example, who like music and no matter what kind. Figure 2 illustrates this.

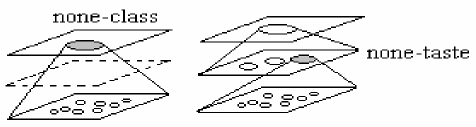


Figure 2. Special classes and clusters in two-layered or three-layered ICN architecture

The upper levels, that is, the sets of classes and clusters are fixed beforehand in the ICN. This can be easily done by the software provider.

Users can decide which cluster they will join, as well as leave. These actions can also be determined by statistic data from individual user actions. More details about the actions of nodes will be illustrated later in this paper. Each class has a unique binary id, so does each cluster.

3.1 Node information

Each node has a unique node id as shown in Figure 3:

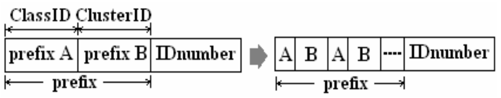
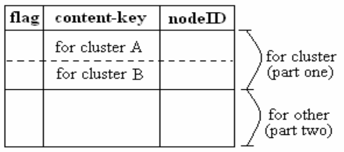


Figure 3. Node id structure

Because one node may belong to different classes and different clusters, each node prefix has a flexible length, as in Figure3 (right).

Each node also keeps a routing table as in Figure 4:



flag	content-key	nodeID
	for cluster A	
	for cluster B	

for cluster (part one)
for other (part two)

Figure 4. Routing table structure

The *Flag* field is reserved for routing table management function extensions, e.g. mark for different ISPs. There are two parts in the nodes' routing table: part one is for locating in the cluster the node is registered and part two is formed by retrieving data from other places. The size for part one is dynamical according to the number of clusters the node belongs to. The second part size is fixed. Assuming N entries for each cluster and M entries for part two, one node that has registered for K different clusters will have a routing table with $NK+M$ entries.

There is a strong connection between caching and routing in DHT topologies. For effective caching, copies of objects should be placed in such a way that routing paths are shortened [14]. While Freenet cares little about this, ICN has a simple but effective cache management mechanism built in each node. This mechanism is explained in the following sections.

3.2 Self clustering and ICN construction

ICN has two or three logical layers in a dynamic environment where nodes can join and leave at any time. On the node level, nodes can cluster and re-cluster.

For a new node, the process to register for one cluster usually takes three steps shown in Figure 5: First the new node *a* must find a node *b* already in the ICN, and download all the routing table information from *b*; Then the new node *a* contacts a node *c* that is already in the cluster node *a* prefers. This procedure will be done by a regular search as described in the following subsection. If the node does not choose any cluster to register, the node will try to contact a node in the *null-class*; Finally, download class id, cluster id and routing table from the contacted node *c*. Node *a*'s id will change adopting the prefix of the same class id and cluster id as node *c*.

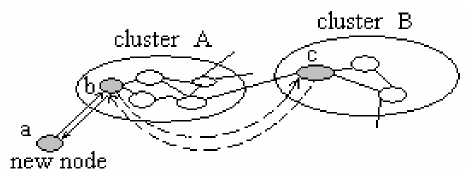


Figure 5. Adding a node

After the node has registered, a function built in each node begins getting statistic data for further control.

There are two typical circumstances this function looks for. If in the beginning the node registered in a cluster it is not really interested in, after some time, the function notes this and removes the corresponding class id and cluster id from the node id. That is, this node will be kicked out of its non-matching interest cluster. This node will change its prefix to point to the *null-class* or the *null-taste*, if it has no other prefixes. In its routing table some entries in part one will move to part two. Another circumstance is the opposite. If a node in the beginning just registers in the *null-class* or the *null-taste*, but its actions like resource sharing and requests show a strong preference, the function will make the node register to the proper cluster following steps 2 and 3 mentioned above.

The control mechanism working together with the routing table management improves local resource usage efficiency, data retrieving efficiency and depresses overhead traffic.

3.3 Retrieving Data

In ICN, before sending a request message, the user must first obtain or calculate the binary id for the file she wants. This file id also contains a class id and a cluster id.

In three-layered ICN, there are three kinds of data retrieving: from the same cluster, from a different cluster in the same class, and from a different class. Similarly, in two-layered ICN, two kinds of data retrieving exist.

When a node requests for some data in the same cluster, chain model search is used as what happens in Freenet. A steepest-ascent hill-climbing search with backtracking is used to locate the objective. Loop detection and a HopsToLive (Freenet's TTL) counter are added to avoid request looping and exhaustive searching. Figure 6 shows a typical sequence of request messages. A request for key 8 is initiated at node A. Node A forwards the request to node B, which forwards it to node C because in B's routing table C is the node that has the closest key to key 8. Node C is unable to contact any other nodes and returns a backtracking "request failed" message to B. Node B then tries its second choice, D. Node D finds key 8 in its routing table and forwards the request to the corresponding node E. The data is returned from E via D and B back to A ending this request sequence. The data is cached on D, B and A. An entry for key 8 is also created in the routing tables of D, B and A.

Data inserts follow a similar strategy to requests in ICN. However, insertion request will never be forwarded to other clusters. It largely decreases the traffic caused by data insertion.

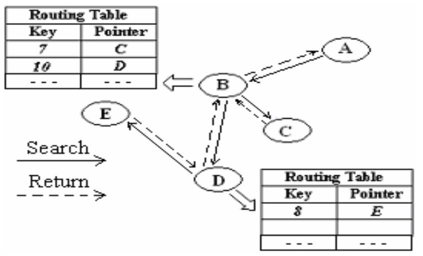


Figure 6. Simplified illustration for chain search process in ICN. All nodes have already registered in one cluster and they are using the proper cluster part in part one of the routing table. In intra cluster search part two is not used.

When a node is requesting an objective that is not in the cluster where this node is, routing between clusters and classes is needed. Classes or clusters in the upper layers are bounded by de Bruijn graph according to their ids.

Figure 7 shows a de Bruijn graph for *k = 2* and *N = 8*, where *k* is the fixed number for outgoing and incoming edges at each class, *N* is the total number of classes. The original graph is shown in [15]. Here we just use it as an example. *N* and *k* can be determined in advance.

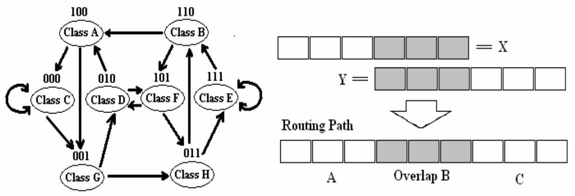


Figure 7. Class/cluster level de Bruijn graph bounding, an example (left) and Routing path in de Bruijn graph (right)

One of the sweet pots for de Bruijn graph is routing. The shortest path routing between any two nodes in a de Bruijn graph follows a greedy procedure in a distributed manner [14]. The algorithm used to find the next hop is quite simple. Suppose X and Y are two class ids and they are two nodes in the de Bruijn graph. If X wants to find the path to Y, it just needs to find the longest overlap between them. By merging A from X, overlap B, and C from Y, X can get a complete path to Y. For example, if class A wants to find class H, the longest overlap of 100 and 011 is 0, so we can get path 10011 that means $100 \rightarrow 001 \rightarrow 011$.

Hence, once the de Bruijn graph has been created in upper levels, shortest path routing can be done in a distributed manner.

The entire process for a node retrieving an objective is described below: The first two steps are common:

1. Generate the key for the objective content with corresponding cluster id and class id;
2. Comparing with all its own cluster ids and class ids, find if the objective is in another cluster, another class or not.

Node now knows clearer what does it want. The following procedures can be different under three circumstances:

- *Circumstance One: The target is in the same cluster as where the requesting node is.* A chain mode search process will be called to locate the content, as what was described earlier. The node will keep a copy in cache along the path while retrieving data successfully. The whole procedure is just like in Freenet.
- *Circumstance Two: The target is not in the cluster but resides in the same class.* The node will first look up in its routing table part two for a "short cut", that is an entry with the node id of the same cluster id as the object. If the "short cut" is found, a request message will be forwarded to this node. Then this node will execute the search-in-cluster process as in circumstance one. If there is no such "short cut", using the properties of the de Bruijn graph, the node knows what are the neighbors (cluster ids) on the shortest path. Hence, it will look for this kind of node id in its routing table part two. If such a node is found, a request message will be forwarded to this node. And the search-in-cluster process is repeated. If there are no such nodes, a request message will be forwarded to the node with the nearest cluster id. This node will carry on the search r. In ICN, when returning the data, the nodes do not cache the copy of the content from a different cluster, instead, just an entry will be added in the routing table.

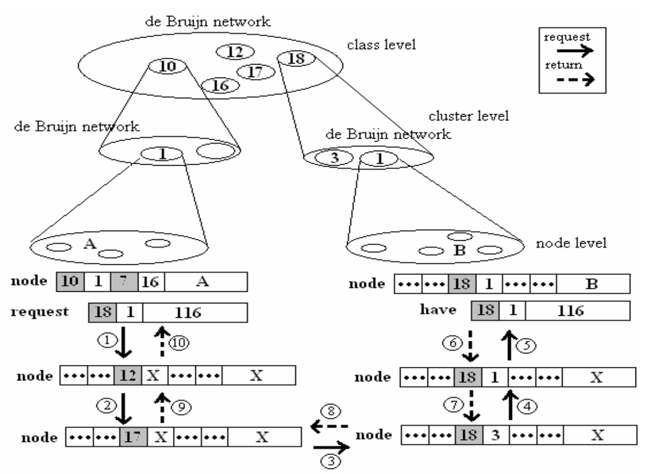


Figure 8. An example: Retrieving data in a difficult class

Circumstance Three: The target is in a different class. To illustrate the procedure, Figure 8 provides an example.

In Figure 8: *Step 1*: Node A wants to find a file in class 18, cluster 1 with content key 116. Node A looks for "short cut" firstly in routing table part two. Since no "short cut" is found, upper level routing is needed. A is in class 10 and class 7. Let us assume that in class de Bruijn network class 10 is nearer to class 18 than class 7. Consequently, A tries to forward the request message to the next hop for 10, say class 16, according to the de Bruijn graph. However, A cannot find any node in its routing table with the class id 16. So, the request message is forwarded to a node with class id 12 that is the next hop in the shortest path in class de Bruijn graph for class 7. *Step 2*: The node with class id 12 looks for a "short cut" in its routing table but fails. The request message is forwarded to a node with the class id 17 that is the next hop on the shortest path in class de Bruijn graph for class 12. *Step 3*: The node with class id 17 looks for "short cut" in its routing table but fails. The request message is forwarded to a node with the class ID 18, which is the next hop on the shortest path in class de Bruijn graph for class 17. Now this node knows that the objective content is in the same class but in a different cluster. This node will execute the search as described under circumstance two. *Step 4*: The request message is forwarded to a node in the same cluster as the object content. *Step 5*: Through searching in the cluster, just like under circumstance one, finally the node that keeps the objective content is found. *Step 6*: Return the data to the upstream node where the request came from, cache a copy of the data and add an entry in its routing table part one. *Step 7*: Return the data to the upstream node where the request came from and add an entry in its routing table part two. This node does not cache a copy because the content belongs to a different cluster. *Step 8*: Return the data to the upstream node where the request came from and add an entry in its routing table part two. This node does not cache a copy because the content belongs to a different class. *Step 9*: Return the data to the upstream node where the request came from and add an entry in its routing table part two. This node does not cache a copy because the content belongs to a different class. *Step 10*: Return the data to the upstream node where the request came from and add an entry in its routing table part two. This node does not cache a copy because the content belongs to a different class.

3.4 Cache Management

Cache management plays a very important role in ICN.

In ICN, nodes also keep the copies of data by using LRU (Least Recently Used) cache in which data items are stored in decreasing order by the time of most recent request. LRU is also used for both parts of the routing table.

Further, ICN caches copies of data with discrimination. A node in ICN just caches the content with the same cluster id as what it has registered for. This procedure is illustrated in the above example from step 6 to 10. Since interest guides the behavior, this caching mechanism will largely improve the efficiency for most requests without exhausting storage resources. The bindings in upper layers are also achieved by caching with discrimination. In a directional de Bruijn graph, the number for downstream clusters or classes is limited. The entries in part two of a routing-table are for contents with downstream cluster or class ids. According to what id the node itself has, the node keeps entries of downstream cluster or class ids with high priority. Hence, most entries in part two provide binding while some existing other entries in part two can provide "shortcuts" for popular content located in some other class to facilitate subsequent requests.

4. Evaluation and analysis

4.1 Routing efficient

When evaluated as a whole, ICN can be seen as a logical tree, like we show in Figure 9. The logical tree architecture gives us a direct view that the overall routing efficiency mainly depends on de Bruijn routing in upper layers and on routing among nodes in the final cluster.

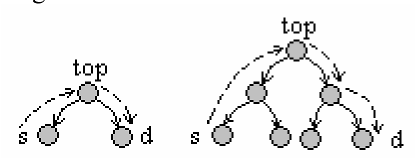


Figure 9. Tree-like architecture in two or three layer ICN

The paper [16] gives an analysis on de Bruijn graph that is the basis for our ICN upper layer routing.

K	de Bruijn	Tries	Chord	CAN	Pastry	Classic butterfly
2	20	--	--	huge	--	31
3	13	40	--	--	--	20
4	10	26	--	1000	--	16
10	6	13	--	40	--	10
20	5	10	20	20	20	8
50	4	8	--	--	7	7
100	3	6	--	--	5	5

Table 1 [12]. Graph diameter for the number of peers N = 10^6 (cells with a dash indicate that the graph does not support the corresponding node degree). k is the fixed number for outgoing and incoming edges. Viceroy [17] and Ulysses [18] are based on an extension of the classical butterfly graph.

From Table 1, we can see that de Bruijn graph is very good in terms of the diameter. Even when the number of classes equals 10^6, or the number of clusters in each class equals to 10^6, we still can construct a graph with a very small diameter. Since the diameter of a graph means the maximum distance for a pair in the network, it provides the upper bound for the required hops in routing. Due to the limitation for the number of classes or clusters, it is highly possible to construct a de Bruijn graph with a very low diameter in the upper layers of ICN.

The diameter for a de Bruijn graph is: $D = log_k N$, where N is the total number of peers. The average distance for a pair in de Bruijn graph follows $\mu_d \approx D-1/(k-1)$, where k is the degree [15][16]. Hence, de Bruijn graph offers low end-to-end upper bounds for the required routing hops if no loops exist.

Considering intra-cluster routing, although the chain model looks very similar to Freenet, except for some caching management, we should note two aspects. First is the number of nodes. Since we are in clusters, the number of nodes is definitely smaller than in global scale Freenet. Second is the request pattern. Since clusters are interest based, requests will be of more regular pattern than those in Freenet. In one cluster, most traffic is for content of the same kind.

We fist check the Freenet search model. The effect of caching policy will be analyzed in the following subsection. Suppose there are totally N_T different files being cached in N nodes for sharing. Each node has a cache room for K files. HopsToLive (Freenet's TTL) is n, which is actually the limitation for searching steps. Let s be the possible number of steps used to search the target. And we suppose search steps are evenly distributed and $N_T - 1 \approx N_T$.

If there are no repeat contents in each node and no search loops, the possibility for hitting the target can be obtained:

$$P(hit) = P(hit \mid s=0)P(s=0) + \ldots + P(hit \mid s=n)P(s=n)$$

$$= \frac{1}{n+1}\left(\frac{K}{N_T} + (1-\frac{K}{N_T})\frac{K}{N_T-K} + (1-\frac{K}{N_T})(1-\frac{K}{N_T-K})\frac{K}{N_T-2K} + \ldots\right.$$

$$\left. + \prod_{i=0}^{n-1}(1-\frac{K}{N_T-iK})\cdot\frac{K}{N_T-nK}\right) = \frac{1}{n+1}\sum_{i=0}^{n}\prod_{j=0}^{i-1}(1-\frac{K}{N_T-jK})\cdot\frac{K}{N_T-iK}$$

In some time snap there are NK files in nodes and N_T is fixed. When $NK > N_T$, some repeat files exist. We define the repeat possibility $P_r(n)$: the possibility for some cached file in the node checked in nth step to be the same as some file in the nodes checked before, that is, in the nodes checked in 0th...$(n-1)$th steps. $P_r(0) = 0$.

$$P_r(1) = 1 - \binom{N_T-K}{K}\Big/\binom{N_T}{K}, \quad P_r(2) = 1 - \binom{N_T-K-K(1-P_r(1))}{K}/\binom{N_T}{K}, \ldots$$

$$P_r(n) = 1 - \binom{N_T - \sum_{i=0}^{n-1}K(1-P(i))}{K}/\binom{N_T}{K}$$

$$P(hit) = \frac{1}{n+1}\left(\frac{K}{N_T} + (1-\frac{K}{N_T})\frac{K(1-P_r(1))}{N_T-K}\right.$$

$$+ (1-\frac{K}{N_T})(1-\frac{K(1-P_r(1))}{N_T-K})\frac{K(1-P_r(2))}{N_T-K-K(1-P_r(1))}$$

$$\left. + \ldots + \prod_{i=0}^{n-1}(1-\frac{K(1-P_r(i))}{N_T-\sum_{j=0}^{i-1}K(1-P_r(j))})\cdot\frac{K(1-P_r(n))}{N_T-\sum_{i=0}^{n-1}K(1-P_r(i))}\right)$$

$$P(hit) = \frac{1}{n+1}\sum_{i=0}^{n}\prod_{j=0}^{i-1}(1-\frac{K(1-P_r(j))}{N_T-\sum_{k=0}^{j-1}K(1-P_r(k))})\cdot(\frac{K(1-P_r(i))}{N_T-\sum_{j=0}^{i-1}K(1-P_r(i))})$$

Now we consider the condition that loops are possible. If the search process hits the target in n steps, then the maximum number of the checked nodes is n, and the maximum number of steps that can be wasted due to loops is $n-2$. That one step is wasted means that one node that has already been checked is checked again. Suppose the number of possible loop is evenly distributed.

Let $P_1(A=a) = P_1(A=a \mid B=b_1)P_1(B=b_1) + P_1(A=a \mid B=b_2)P_1(B=b_2)$ denote the possibility for hitting target in a steps with b steps wasted because of loops. Let $P_1(a \mid b)$ denote $P_1(A=a \mid B=b)$ for simplicity and S_n denote the possibility for hitting the target in nth step. Then we have:

$S_0 = P_1(0 \mid 0), S_1 = P_1(1 \mid 0) \quad S_2 = P_1(2 \mid 0)$,

$$S_3=(P_1(3|0)\frac{1}{2}+P_1(3|1)\frac{1}{2})=\frac{1}{2}(P_1(3|0)+P_1(2|0)),\cdots$$

$$S_n=\frac{1}{n-1}\sum_{i=2}^{n}P_1(i|0).$$ Hence we have:

$$P(hit)=\frac{1}{n+1}(P_1(1|0)+P_1(2|0)+\sum_{i=3}^{n}\frac{1}{i-1}\sum_{j=2}^{i}P_1(j|0))$$

$$P_1(n|0)=\prod_{i=0}^{n-1}(1-\frac{K(1-P_r(i))}{N_T-\sum_{j=0}^{i-1}K(1-P_r(j))})\cdot\frac{K(1-P_r(n))}{N_T-\sum_{i=0}^{n-1}K(1-P_r(i))})$$

Consider our two-layered ICN model, where the top layer is classes bound by a de Bruijn graph: Suppose *N* nodes in *k-connectivity* de Bruijn graph, which has Q interest classes. Its diameter is *2*. Request is from class A to class B. We do not consider the effects brought by short-cuts.

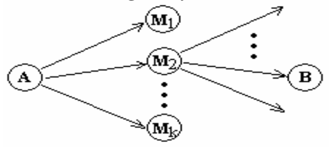


Figure 11. Request from class A to a proper middle node, then from the middle node to class B

In the nodes in A, M$_1$, M$_2$,…M$_k$, we are not looking for contents, but proper id. And when request is forwarded to B, search is carried out just like above. Now we consider the steps needed for request being forwarded from A to a middle class and from the middle class to B. The number of entries in routing table part two is *T'*. The probability for that from A to the middle class in one step is P_M. *S1* denotes the step needed from A to a middle class.

$$\overline{P}_M=\frac{(k-1)^{T'}}{k^{T'}}=(1-\frac{1}{k})^{T'},\quad P_M=1-\overline{P}_M$$

After considering the circumstance with loops:

$$P_{S1=n}=\frac{1}{n-1}\sum_{i=2}^{n}P_1(i|0),3\le n\le TTL$$

$$P_1(n|0)=\overline{P}_M^{\,n-1}P_M,1\le n\le TTL$$

The probability that the entries of our middle class node do not contain an id for B:

$$\overline{P}_B=\frac{(k-1)^{T'}}{k^{T'}}=(1-\frac{1}{k})^{T'},\quad \text{hence } P_B=1-\overline{P}_B$$

S2 denotes the step needed from the middle class to B.
After considering the circumstance with loops:

$$P_{S2=n}=\frac{1}{n-1}\sum_{i=2}^{n}P_2(i|0),3\le n\le TTL$$

$$P_2(n|0)=\overline{P}_B^{\,n-1}P_B,1\le n\le TTL$$

S3 denotes the step needed to hit the target in class B.
Combining the above we get:

Hit target in 2 steps: S1=1, S2=1, S3=0;
Hit target in 3 steps: S1=1, S2=1, S3=1;
S1=1, S2=2, S3=0;
S1=2, S2=1, S3=0;
…
Hit target in n steps: S1: from 0 to n-1
S2: from 1 to n-S1
S3: n-S1-S2

P_n denotes the probability to hit the target in n steps in ICN:

$$P_n=\frac{1}{1+n}\sum_{i=2}^{n}\frac{2}{(i-1)i}\sum_{j=1}^{n-i}P_1(i|0)P_2(j|0)P_3(n-i-j|0)$$

Where

$$P_3(n|0)=\prod_{i=0}^{n-1}(1-\frac{K(1-P_r(i))}{N_T-\sum_{j=0}^{i-1}K(1-P_r(j))})\cdot\frac{K(1-P_r(n))}{N_T-\sum_{i=0}^{n-1}K(1-P_r(i))})$$

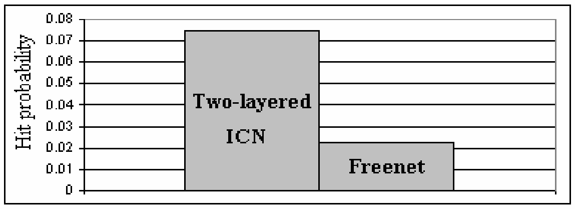


Figure 12. In two-layer ICN: Number of different files is 10000, TTL is 10, file caching room is 250, the number of entries in routing table part two is 100, number of classes is 10, connectivity is 4. In Freenet, the number of different files is 10000, TTL is 10, caching room for files is 250. File repeat probability in both is 0.

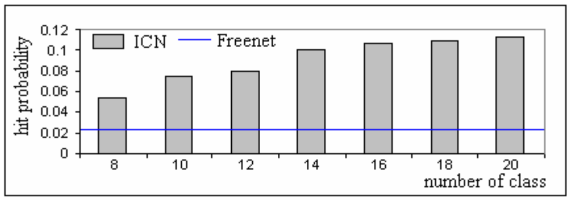


Figure 13. In two-layer ICN: Number of different files is 10000, TTL is 10, file caching room is 250, the number of entries in routing table part two is 100, number of classes is from 8 to 20, connectivity is 4. In Freenet, the number of different files is 10000, TTL is 10, caching room for files is 250. File repeat probability in both is 0.

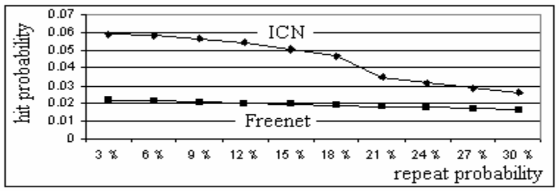


Figure 14. In two-layer ICN: Number of different files is 10000, TTL is 10, file caching room is 250, the number of entries in routing table part two is 100, number of classes is from 8 to 20, connectivity is 4. In Freenet, the number of different files is 10000, TTL is 10, caching room for files is 250. File repeat probability in both is from 3% to 30%. File repeat probability in ICN means that in each class.

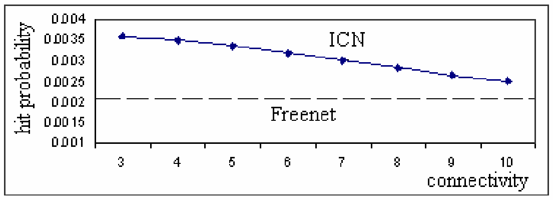


Figure 15. In two-layer ICN: Number of different files is 10000, TTL is 10, file caching room is 250, entries number of routing table part two is 100, number of class is from 8 to 20,

connectivity is from 3 to 10. In Freenet, the number of different files is 10000, TTL is 10, caching room for files is 250.

From Figures 12-15, the advantages of ICN when evaluating hit probability are clear. Furthermore, our results show some trends for parameters changes.

4.2 Effect from request model

We use two models to simulate the impact of interest pattern on data retrieving. One is time a shifting zipf model and the other is a hot-cold model. The results given in this subsection are from our Freenet simulator [19]. These results can help us with the analysis of the advantages of ICN.

A simple description of data that follow a Zipf popularity distribution is that 1) a few items of content are requested very frequently 2) there is a medium number of items with middle-of-the-road frequency of requests); 3) a huge number of items that are requested very rarely[20]. We further put a time shifting function on it. We allow new content item to join the ranking and the old ones to change their ranking. So in our simulation, rank is not static while the whole still follows a zipf distribution. However, our simulation does not show the preference of ranking for the newly inserted items. A hot-cold model simply means that about 10% of content items attract about 90% of requests. Also some interest shifts exist in our simulation.

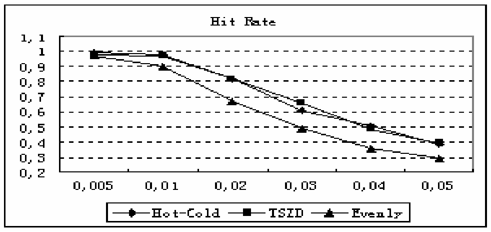


Figure 16. Hit Rate will decrease while request messages are generated more often. The x scale is the probability that each node will generate a request.

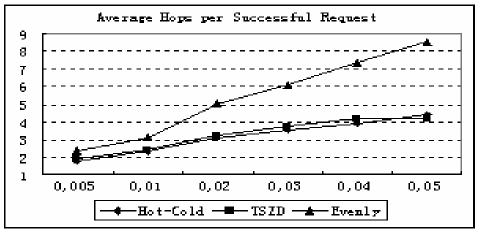


Figure 17. Average Hops per Successful Request will increase while request messages are generated more often. The x scale is the probability that each node will generate a request.

Figures 16 and 17 show that Hit Rate and Average Number of Hops per Successful Request will be quite different for different request model. Especially when the request traffic is larger, the difference is very clear. We believe time shifting zipf model and hot-cold model match the real world better than even distribution.

Results show that Freenet performance is quite sensitive to the request models due to it caching files without discrimination. In ICN, we use the nearly same strategy in the cluster or class where the target is. Hence, our performance in the final cluster is also sensitive. However, in Freenet, the caching is global while in ICN the caching is localized because of the policy that ICN nodes will not cache the files from other clusters.

This localization brought by ICN cache management policy has two main advantages: 1) like Freenet, it will facilitate search for those popular targets. Even for the request from other clusters, because nodes in the middle cluster have the proper routing entry with high priority, the target can be found with high efficiency; 2) better protection for those contents in other clusters. Because LRU is used in every node, localized caching will definitely alleviate the impact of popular contents from other clusters. When looking for some less popular contents, it will give a better performance. It also provides a more user-friendly environment. As a music fan, you may be happy for some new movies, but these movies should not hamper your search for music so much.

4.3 Robustness

As a Freenet alike network shows a surprisingly robust behavior under quite a large portion of nodes failing [8], the robustness of ICN intra cluster networks can be guaranteed.

When considering the inter class or inter cluster communication, we check the architecture of the upper levels in ICN. The cluster level and the class level are based on de Bruijn graph.

A smaller cluster coefficient means that in a peer-to-peer network a request can reach more nodes within a certain number of hops and that the network can provide a more fault-resilient environment where a simultaneous collapse of several nodes does not separate the graph into disjoint components [16]. The cluster coefficient of a de Bruijn graph is *(k-1)/N*, which is really small.

Further, we note that in Figure 7, there are two self cycles in nodes 000 and 111. These cycles do not exist in ICN. In ICN, to be more precise, chain-linked de Bruijn graphs are used. Consider node (h,h,…h), $h \in \Sigma$, with a self-loop. A chain-linked de Bruin graph has directed links $(h,h,..h) \rightarrow (g,g,..g)$, for all $h \in \Sigma$ and $g=(h+1)$ mod k. Research in [21] proved that chain-linked de Bruijn graphs are k-node connected. A k-node-connected graph can tolerate the failure of any $k-1$ nodes without becoming disconnected. And after any $k-1$ nodes have failed, the diameter of a k-node-connected graph is at most D+1, where D is the diameter before nodes failed.

Analysis in [16] shows that the de Bruijn graph can further offer optimal resilience, large bisection width, and good node expansion that guarantees very little overlap between parallel paths to any destination.

In ICN, node failure or leave in upper level de Bruijn graph means that nearly all nodes in some cluster or in some class malfunction or leave, which is an event with a rather small probability. However, even when this happens, the de Bruijn architecture can still work very well.

4.4 Commercial view

The ICN scheme is great for classifying users by their interest. Therefore, content providers immediately can segment the users very efficiently and can target their new offerings in the best way. No need to waste efforts in pushing stuff to people who are very unlikely to buy your goods. On the other hand, if the content owner can target his new offering to users of a cluster, it is likely that the promotion activities are cost effective.

5. Conclusion

ICN shows many advantages as a p2p architecture. Although there are logical levels in ICN, the essence is still node-based distribution. We simply add some prefix to nodes and then make them cluster to facilitate retrieving data. The ICN architecture is different from a traditional tree shape hierarchical network. In particular, ICN has no single vulnerable spot. Upper logical levels are connected following a de Bruijn graph improving the routing efficiency and network robustness. Cache management mechanisms not only support ICN structure but also make searching more efficient in clusters using routing table part one and provide possible shortcuts using routing table part two when requesting files from other clusters or classes.

The drawback of ICN is that for each node in ICN the routing table grows larger when a node registers for more interest clusters or classes.

6. Future work

Although ICN is based on the graph theory and good performance can be expected directly, a proper human interest or behavior model with strong statistical support is needed for evaluating the whole system more accurately.

Further more, we see a need to study new routing table and cache management mechanisms for further improvement.

7. Reference

[1] Napster. http://www.napster.com
[2] S. Ratnaswamy, P. Francis, etc., "A scalable content-addressable network"
[3] J. Kubiatowicz, D. Bindel, Y. Chen, etc., "Oceanstore: An architecturefor global-scale persistent storage", Proceedings of the ASPLOS Nov. 2000
[4] Gnutella. http://www.gnutella.co.uk
[5] I. Stoica, R. Morris, etc., "Chord: A peer-to-peer lookup service for internet applications", ACM SIGCOMM,2001
[6] Y. Chu, S. Rao, etc., "A Case for end system multicast. Proceedings of ACM Sigmetrics", 2000
[7] Yallcast. http://www.yallcast.com
[8] I. Clarke, O. Sandberg, etc., "Freenet: A distributed anonymous information storage and retrieval system in designing privacy enhancing technologies". LNCS, 2001
[9] A. Rowstron, P. Druschel, "Pastry: Scalable, Decentralized Object location and routing for large-scale peer-to-peer system", IFIP/ACM ICDS, 2002
[10] Reading Club Survey. http://ihouse.hkedcity/
[11] AFW Cultureral Survey. http://geoctities.
[12] Clips, "The Gnutella Protocol Specification", http://www.clip2-.com, 2000
[13] Ajay chander, Steven Dawson, etc., "NEVRLATE: Scalable Resource Discovery", Procedings of the 2nd CCGRID'02
[14] Gurmeet Singh Manku, "Routing Networks for Disteributed Hash Table", ACM PODC'03, 2003
[15] K.N.Sivarajan, R. Ramaswami, "Lightwave Networks Based on de Bruijn Graph", IEEE/ACM Trans. on Networking, vol. 2, no.1, 1994
[16] Dmitri Loguinov, Anuj Kumar, "Graph-Theoretic Analysis of Structured Peer-to-Peer System: Routing Distances and Fault Resilience". SIGCOM, 2003
[17] D.Malkihi, M. Naor, etc., "Viceroy: A Scalable and Dynamic Emulation of the Butterfly", ACM PODC, 2002
[18] J.Xu, A. Kumar,etc., "On the Fundamental Tradeoffs between Routing Table Size and Network Diameter in Peer-to-Peer Networks", IEEE JSAC, Nov. 2003
[19] Raimo Kantola, "Peer to Peer and Spam in The Internet", Seminar Report, Helsinki University of Technology, 2004
[20] Zipf's Law. http://www.cpe.ku.ac.th/~arnon/
[21] D.Z.Du, D.F.Hsu "On Connectivity of Consecutived Digraphs", Discrete Mathematics, vol. 257, no. 2-3, 2002