Building a Scalable P2P Network with Small Routing Delay*

Shiping Chen[1,2], Yuan Li[2], Kaihua Rao[2], Lei Zhao[2], Tao Li[2], and Shigang Chen[3]

[1] Network Center, University of Shanghai for Science and Technology, Shanghai, 200093, China
[2] Department of Computer Engineering, University of Shanghai for Science and Technology, Shanghai, 200093, China
[3] Department of Computer and Information Science and Engineering, University of Florida, USA
chensp@usst.edu.cn

Abstract. Most existing P2P networks route requests in $O(kN^{1/k})$, $O(\log N)$, $O(\log N/\log k)$ hops, where N is the number of participating nodes and k is an adjustable parameter. Although some can achieve $O(d)$-hop routing for a constant d by tuning the parameter k, the neighbor locations however become a function of N, causing considerable maintenance overhead if the user base is highly dynamic as witnessed by the deployed systems. This paper explores the design space using the simple uniformly-random neighbor selection strategy, and proposes a random peer-to-peer network that is the first of its kind to resolve requests in d hops with a chosen probability of $1-c$, where c is a constant. The number of neighbors per node is within a constant factor from the optimal complexity $O(N^{1/d})$ for any network whose routing paths are bounded by d hops.

Keywords: Peer-to-Peer Networks, Randomized Topology, Routing Delay.

1 Introduction

Peer-to-peer (P2P) systems have many applications in data sharing, notification services, data dissemination, directory lookup, software distribution, and distributed indexes. Because data may be kept at any node, a fundamental problem is to efficiently locate the node that stores a particular data item. Napster uses a centralized directory service. Gnutella [1] and KaZaA [2] rely on flooding-based search mechanisms, which cause tremendous communication overhead for large systems [3,4,5,6].

To solve the scalability problem, many P2P proposals use distributed hash tables (DHT) to uniformly distribute the responsibility of data location management to all nodes. An identifier is associated with each data item, and each node is responsible for storing a certain range of identifiers together with the corresponding data items or

* This research was Supported by the National Natural Science Foundation of China under Grant No. 60573142, The Shanghai Leading Academic Discipline Project under Grant No. T0502.

their locations (addresses). DHT provides a basic function, $lookup(id)$, which maps an arbitrary identifier to the responsible node. To implement such a function, an overlay P2P network is formed among the participating nodes. When a lookup request is issued, the request will be routed to the responsible node via the P2P network. In a highly-dynamic environment where nodes frequently join and depart, the maintenance overhead for the overlay P2P network is a major design concern [7]. A recent survey on different types of P2P networks can be found in [17].

When constructing a P2P network, there exists a fundamental space-time tradeoff between the number of neighbors (i.e., the size of the routing table) and the network diameter (i.e., the length of the routing path) [8]. Many P2P networks have an adjustable parameter (k) that can be tuned for different space-time tradeoffs. For example, if $k = \log N$, both time and space complexities of CAN become $O(\log N)$, where N is the number of nodes in the system. For all P2P networks, however, the maintenance overhead is minimized when k is a constant — instead of a function of N that changes continuously as nodes join/depart.

PRR [9] and Pastry [10] require $O(k\frac{\log N}{\log k})$ neighbors per node and route in $O(\frac{\log N}{\log k})$ hops with high probability. In the following, we shall omit "with high probability" as it is true for most complexities to be described. Tapestry [11] and Chord [12] require $O(\log N)$ neighbors and route in $O(\log N)$ hops. CAN [13] requires $O(k)$ neighbors and route in $O(kN^{\frac{1}{k}})$ hops.

The first asymptotically-optimal system is Viceroy [14], which requires seven neighbors per node and routes in $O(\log N)$ hops. Koorde [15], and Manku [16], achieve asymptotical optimality with $O(k)$ neighbors and $O(\frac{\log N}{\log k})$ routing hops, where [16] assumes $k = O(polylog(n))$.

In the family of P2P networks, one important member is much less investigated, i.e., one with $O(N^{\frac{1}{d}})$ neighbors per node and d routing hops, where d is a constant. Such a network is appealing in practice because of its small routing delay, which does not grow with respect to the size of the network. Each routing hop in a P2P network requires a message to travel end-to-end from one node to another, likely crossing the Internet. Given the prevalence of inexpensive memory, it is often desirable to trade more neighbors (space) for shorter routing paths (delay). For increased number of neighbors, the main problem is not the space requirement, but the complexity for maintaining the neighboring relationship [7]. This is particularly true for structured networks such as PRR, Pastry, and randomized Chord, where the neighbors of a node x are required to match the top i digits of x and differ at the $(i+1)th$ digit, for $i \in [1 ... \log_k N]$, where k is the base of the digits. By choosing $k = N^{\frac{1}{d}}$, these systems achieve $O(d)$ routing hops with $O(dN^{\frac{1}{d}})$ neighbors. However, the neighbor locations are now a function of N because the base k is related to N. As N changes, the base of the digits ($N^{\frac{1}{d}}$) changes, which can make many existing neighbors no longer valid, causing considerable maintenance overhead.

One solution for reducing maintenance overhead is to use random neighbors, which require little maintenance. A node can take any other nodes as its neighbors based on certain probability distribution. Among the random P2P networks [14,,16], [16] have an adjustable parameter k, which must be a polylog function of N in order for their complexities to hold. For NoN routing [16], $k = O(polylog(N))$. None can achieve constant routing distance by adjusting k.

This paper proposes a new random P2P network that combines arbitrary neighbor selection, typically used only in unstructured P2P networks, with a DHT (distributed hash table) ring. It is the first of its kind to resolve requests in no more than d hops with probability $1-c$, where d and c are two configurable constants. In more conventional terms, choosing a small value (e.g., 10^{-10}) for c, the system resolves an arbitrary request in d hops with high probability (e.g., $1-10^{-10}$). There is a small probability c that a request is not resolved in d hops. When it does happen, a slower routing path will be taken, which guarantees to find the responsible node. The number of neighbors per node is $O((-\ln c)^{1/2} d N^{\frac{1}{d}})$. Random neighbors are easy to manage. When nodes join or depart, the random neighbors of all other nodes remain unchanged. Without sacrificing the performance, a node increases (or decreases) its number of random neighbors only when N doubles (or halves). Note that the location of any particular neighbor is independent of N.

In Appendix A we prove that, for routing paths to be bounded by d hops, the lower bound on the number of neighbors is $\Omega(N^{\frac{1}{d}})$. Therefore, the space complexity of the proposed random P2P network is within a constant factor $(-\ln c)^{1/2} d$ from the optimal.

The rest of the paper is organized as follows. Section 2 defines the model, notations and performance metrics. Section 3 proposes a random peer-to-peer network. Section 4 presents the simulation results. Section 5 shows the time complexity and the space complexity. Section 6 draws the conclusion.

2 Model, Notations and Performance Metrics

Each data item is mapped to an m-bit identifier by a hash algorithm. The whole ID space can be viewed as a modulo-2^m circle, where the next identifier in the circle after the largest value •2^m-1• is zero. Consider N participating nodes. Each node is assigned an identifier by hashing its address or domain name. When the node identifiers are marked on the ID space, they split the circle into N segments. A node x is responsible for the segment (denoted as $seg(x)$) that immediately follows its node identifier. The nodes that are responsible for the adjacent preceding (or following) segments are called the predecessors (or successors) of x. The location information about a data item is stored at x if the identifier of the item belongs to $seg(x)$.

When a user queries for a data item whose identifier is id, she submits a lookup request(id), which is routed through an overlay network to the node that is responsible for the identifier, denoted as $node(id)$. The node subsequently returns the data location to the user. The performance/overhead tradeoff achieved by the routing algorithm is fundamentally determined by the structure of the overlay topology.

Table 1. Notations

N	number of nodes in a peer-to-peer network
m	number of bits in an identifier
x, y, z, w	arbitrary nodes in a peer-to-peer network
$seg(x)$	segment of identifiers that x is responsible for
id	arbitrary identifier to be queried
$node(id)$	node that is responsible for id
S_x	set of sequential neighbors of x
R_x	set of random neighbors of x
$sup\_seg(x)$	segment of identifiers that $S_x + \{x\}$ is responsible for
s	number of sequential neighbors
r	number of random neighbors
$1 - P_d$	probability for a request to be resolved in d or less hops
$1 - c$	target probability of resolving a request in d or less hops
RP2P(d,c)	random peer-to-peer network that resolves a request in d or less of hops with a probability of at least $(1-c)$

The notations defined above and later in the paper are listed in Table 1 for quick reference. We evaluate the performance of a peer-to-peer system based on the following metrics.

1. time complexity: the maximum number of hops that a request(id) must travel in the overlay topology before reaching $node(id)$
2. space complexity: the maximum storage that a node is used to keep the neighbor information

The issues of load balancing [18,19], proximity and locality [20], security [21], pricing, etc., are beyond the scope of this paper.

3 Random Peer-to-Peer Network (RP2P)

Given two constants d and c, our goal is to develop a peer-to-peer network whose time and space complexities are $O(d)$ and $O(N^{\frac{1}{d}})$, respectively. We start with an abstract description of the system. We then present some analytical results and discuss the protocols/algorithms that realize the system.

For all above complexities in the forms of $O(N^{\frac{1}{d}})$, we have omitted factors that are functions of d and c. These factors will be shown in the detailed description of the system.

3.1 Overlay Topology

Each node knows a set of neighbors that it will directly communicate with. There are two types of neighbors, as shown in Figure 1, where the circle represents the ID space.

random neighbors: A node x takes a number of randomly selected nodes as its random neighbors, denoted as R_x .

sequential neighbors: A node x takes a number of predecessors and a number of successors as its sequential neighbors, denoted as S_x . The combination of the segments that $S_x + \{x\}$ are responsible for is denoted as $sup\_seg(x)$, which is called the super segment of x .

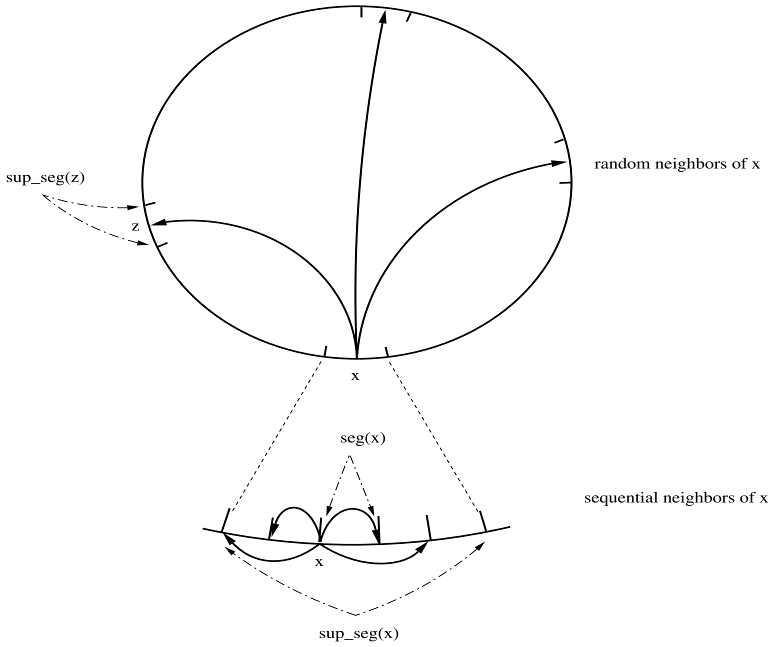

Fig. 1. Random neighbors and sequential neighbors of x

In the example of Figure 1, x has three random neighbors and four sequential neighbors. A node is required to store the following information about its neighbors.

- For each sequential neighbor $y \in S_x$, it uses two integers to store the neighbor's segment, $seg(y)$. Combining all these segments, x also knows its super segment, $sup\_seg(x)$.
- For each random neighbor $z \in R_x$, it uses two integers to store the neighbor's super segment, $sup\_seg(z)$.

The above information is learned from the neighbors. The space complexity for storing the information is equal to the number of neighbors. when x receives a request whose identifier belongs to $sup\_seg(x)$, it knows immediately which node (a sequential neighbor or itself) is responsible for the identifier. On the other hand, if the identifier belongs to the super segment of a random neighbor z, x should forward the request to z.

3.2 Routing Algorithm

When a node x receives a request(id), it processes the request by the following algorithm. Suppose the request carries the address of the node that originates the request.

```
RP2P_Routing( $id$ )
1.  if $id \in seg(x)$ then
2.      process request and send result to original requester
3.  else if $\exists y \in S_x, id \in seg(y)$ then
4.      forward the request to $y$
5.  else if $\exists z \in R_x, id \in sup\_seg(z)$ then
6.      forward the request to $z$
7.  else
8.      forward the request to all random neighbors
```

A few routing examples are given in Figure 2.

zero-hop case: It takes zero hop to resolve a request if $id \in seg(x)$, as shown by the first plot in the figure and implemented by Lines 1-2 of the algorithm.

one-hop case: It takes one hop if $id \in seg(y), \exists y \in S_x$, as shown by the second plot in the figure and implemented by Lines 3-4 of the algorithm.

two-hop case: It takes two hops if $id \in sup\_seg(z), \exists z \in R_x$, as shown by the third plot in the figure and implemented by Lines 5-6 of the algorithm.

Three-hop case: It takes three or more hops otherwise, as shown by the last plot in the figure and implemented by Lines 7-8 of the algorithm.

For the first three cases, x knows for sure which is the next node to forward the request. For the last case, x has no clue about the next node. Hence, it broadcasts the request to all random neighbors. To restrain the broadcast overhead, we introduce a TTL field in the request message such that the request can only travel d or less hops and allows up to $d-2$ levels of broadcast (to random neighbors). As illustrated in the figure, the last two hops do not require broadcast as the node receiving the request has enough information to determine whether two more hops can reach $node(id)$ and if so, which is the next node to forward the request.

Below we give a basic analytical result. Suppose each node has s sequential neighbors and r random neighbors. To simplify the analysis, assume the nodes are responsible for equal-sized segments of the ID space. We will show that the analytical results with this assumption match very well with the simulation results without this

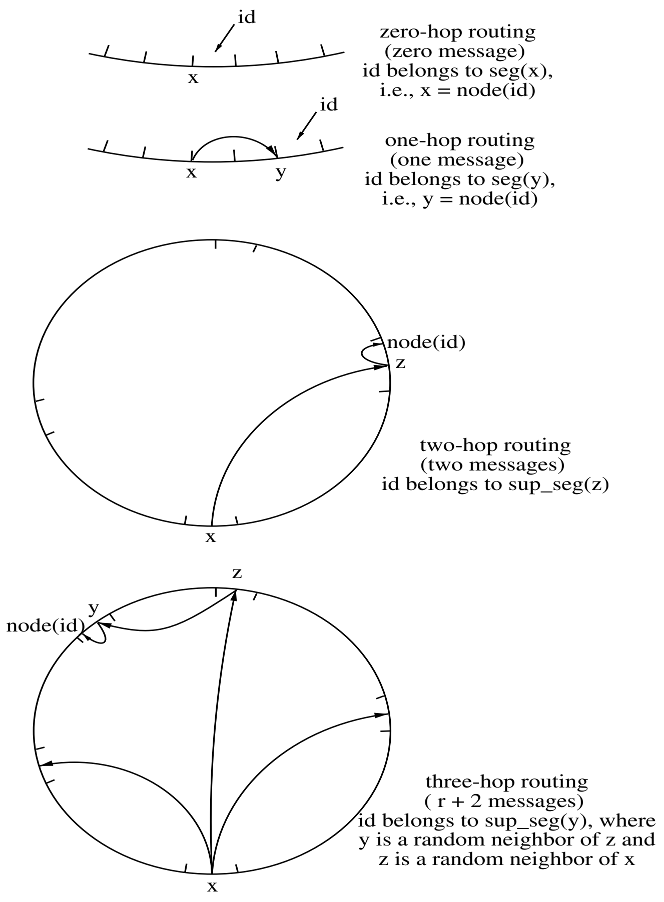

Fig. 2. Routing examples

assumption. Let P_d be the probability for request(id) to NOT reach $node(id)$ in d or less hops. The following upper bound of $P_d, d \geq 2$ is proved in Appendix B.

$$P_d \prec (1 - \frac{s}{N})^{r^{d-1}} \tag{1}$$

We will demonstrate shortly that, by appropriately choosing the values of s and r , a request can be resolved in d or less hops with a chosen probability (e.g., $1-10^{-10}$).

3.3 Determining Appropriate Values for s and r

Consider an integer $d \geq 2$ and a small constant $c \in (0..1)$. We prove that, if $s = r = kN^{\frac{1}{d}}$ where $k = (-ln c)^{\frac{1}{d}}$, then $P_d < c$. By (1), we have

$$P_d < (1 - \frac{kN^{\frac{1}{d}}}{N})^{(kN^{\frac{1}{d}})^{d-1}} \qquad (2)$$

Define the following quantity.

$$q = (\frac{N}{k^{\frac{d}{d-1}}})^{\frac{d-1}{d}}$$

Rewrite (2) as below.

$$P_d < (1 - \frac{1}{q})qk^d$$

$(1-\frac{1}{q})^q$ is a monotonically-increasing function with respect to q, and $limit_{q\to\infty}(1-\frac{1}{q})^q = \frac{1}{e}$, where e is the base of natural logarithm. Hence, we have

$$\begin{aligned} P_d &< (\frac{1}{e})^{k^d} \\ &= (\frac{1}{e})^{(-ln\,c)^{\frac{1}{d}d}} \\ &= c \end{aligned}$$

Let RP2P(d,c) be a random peer-to-peer network where each node has $(-ln\,c)^{\frac{1}{d}} N^{\frac{1}{d}}$ sequential neighbors and the same number of random neighbors.[1] As an example, when $d=3$, it becomes RP2P(3, c). Suppose each request carries a TTL field whose initial value is d . We modify the routing algorithm such that the longest routing path has no more than d hops.

```
RP2P_Routing_TTL( $id$ )
1.    decrease the TTL of the request by one
2.    if $id \in seg(x)$ then
3.      process request and send result to original requester
4.    else if $\exists y \in S_x, id \in seg(y)$ then
5.      forward the request message to $y$
6.    else if $\exists z \in R_x, id \in sup\_seg(z)$ then
7.      forward the request message to $z$
8.    else if TTL of the request $\geq 2$ then
9.      forward the request to all random neighbors
10.   else
11.     discard the request
```

Based on the previous analysis, we have the following theorem.

[1] If $c = 10^{-10}$ and $d = 3$,then $(-ln\,c)^{\frac{1}{d}}$ =2.8.

Theorem 1. The probability for RP2P(d,c) to resolve a request in d or less hops is larger than $1-c$, where $d \geq 2$ and $c \in (0...1)$.

4 Simulation Results

Our simulation results match very well with the analysis. We simulated RP2P(3, c) on networks of 1000,10000, and 100000 nodes, respectively. The simulation was repeated for different values of c. The results are shown in Table 2. The column of c is the target failure probability. The column of s ,r is the number of sequential (random) neighbors. The column of P_3 is the measured probability of NOT resolving a request in 3 or less hops. P_3 is always better (smaller) than the target value c. That is because our analysis made a conservative simplification when using (5) to derive the upper bound of P_d in Appendix B.

Table 2. Simulation results for RP2P(3, c)

	N **= 1,000**		N **= 10,000**		N **= 100,000**	
c	s,r	P_3	s,r	P_3	s,r	P_3
1.0e-1	13	6.8e-2	28	8.8e-2	61	1.1e-1
1.0e-2	16	9.4e-3	35	1.1e-2	77	9.1e-3
1.0e-3	19	4.3e-4	41	7.4e-4	88	8.6e-4
1.0e-4	20	1.4e-4	45	7.1e-5	97	8.1e-5
1.0e-5	22	7.7e-6	48	9.2e-6	104	9.4e-6
1.0e-6	23	1.5e-6	51	8.9e-7	111	8.7e-7
1.0e-7	25	3.5e-8	54	7.0e-8	117	8.4e-8

5 Complexities of RP2P(d,c)

The maximum number of hops that a request will travel in RP2P(d,c) is d, and the time complexity is thus $O(d)$. The number of neighbors per node is $r+s=2(-\ln c)^{\frac{1}{d}}N^{\frac{1}{d}}$, and the space complexity is thus $O((-\ln c)^{\frac{1}{d}}N^{\frac{1}{d}}$.

6 Conclusion

This paper designs a random peer-to-peer network with neighbor nodes selected uniformly at random. The network is the first of its kind to resolve requests within a constant number of hops with high probability. A key advantage is the ease of neighbor management when nodes join/depart. The time and space complexities of the proposed network are $O(d)$ and $O((-\ln c)^{\frac{1}{d}}N^{\frac{1}{d}}$, respectively. We conduct comprehensive analysis to derive the properties of the systems. Our simulation results match with the analytical results.

References

1 Gnutella: Gnutella, http://gnutella.wego.com
2 KaZaA: KaZaA, http://www.kazaa.com
3 Ritter, J.: Why Gnutella can't Scale. No, Really, `http://www.tch.org/gnutella.html`
4 Ripeanu, M., Iamnitchi, A., Foster, I.: Mapping the Gnutella Network. IEEE Internet Computing Journal, Special Issue on Peer-to-Peer Networking 6(1) (2002)
5 Sen, S., Wang, J.: Analyzing Peer-to-Peer Traffic across Large Networks. In: ACM SIGCOMM Internet Measurement Workshop (August 2002)
6 Saroiu, S., Gummadi, K.P., Dunn, R.J., Gribble, S.D., Levy, H.M.: An Analysis of Internet Content Delivery Systems. In: Proc. of the 5th Symposium on Operating Systems Design and Implementation (OSDI) (December 2002)
7 Ratnasamy, S., Shenker, S., Stoica, I.: Routing Algorithms for DHTs: Some Open Questions. In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, vol. 2429, Springer, Heidelberg (2002)
8 Xu, J.: On the Fundamental Tradeoffs between Routing Table Size and Network Diameter in Peer-to-Peer Networks. In: Xu, J. (ed.) Proc. of IEEE INFOCOM 2003 (April 2003)
9 Plaxton, C., Rajaraman, R., Richa, A.: Accessing Nearby Copies of Replicated Objects in a Distributed Environment. In: Proc. of ACM Symposium on Parallelism in Algorithms and Architectures (SPAA) (June 1997)
10 Druschel, P., Rowstron, A.: Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. In: Proc. of 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001) (November 2001)
11 Zhao, B., Kubiatowicz, J., Joseph, A.: Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing, Tech. Rep. UCB/CSD-01-1141, University of California at Berkeley, Computer Science Department (2001)
12 Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. In: Proc. of ACM SIGCOMM 2001 (August 2001)
13 Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A Scalable Content-Addressable Network. In: Proc. of ACM SIGCOMM 2001 (August 2001)
14 Malkhi, D., Naor, M., Ratajczak, D.: Viceroy: A Scalable and Dynamic Emulation of the Butterfly. In: Proc. of ACM PODC 2002 (July 2002)
15 Kaashoek, F., Karger, D.R.: Koorde: A Simple Degree-Optimal Hash Table. In: Kaashoek, M.F., Stoica, I. (eds.) IPTPS 2003. LNCS, vol. 2735, Springer, Heidelberg (2003)
16 Manku, G.S.: Routing Networks for Distributed Hash Tables. In: Proc. of 22nd ACM Symposium on Principles of Distributed Computing (PODC) (June 2003)
17 Risson, J., Moorsa, T.: Survey of Research towards Robust Peer-to-Peer Networks: Search Methods. Journal of Computer Networks 55 (2006)
18 Joung, Y.-J., Yang, L.-W., Fang, C.-T.: Keyword search in DHT-based peer-to-peer networks. IEEE Journal on Selected Areas in Communications 25 (2007)
19 Li, Z., Xie, G.: A Distributed Load Balancing Algorithm for Structured P2P Systems. In: Proc. of the 11th IEEE Symposium on Computers and Communications (June 2006)

20 Ferreira, R.A., Jagannathan, S., Grama, A.: Locality in structured peer-to-peer networks. Journal of Parallel and Distributed Computing 66 (2006)

21 Navabpour, S., Nejad, N.F., Abbaspour, M., Behzadi, A.: Secure Routing in Structured Peer to Peer File-Sharing Networks. In: Proc. of International Conference on Communications and Networking in China (ChinaCom 2006) (October 2006)

Appendix A. Number of Neighbors Per Node in Networks of Constant Diameter

Theorem 1: The average nodal degree must be $\Omega(N^{\frac{1}{d}})$ for an N-node network with diameter d .

Proof: For a network with diameter d, starting from an arbitrary node, we can reach all nodes by a breadth-first search tree of d levels in depth. Let x be the average nodal degree. The number of nodes in the tree is $N = O(\sum_{i=0}^{d} x^i) = O(x^d)$. In order for $N = O(x^d)$ to hold, it is required that $x = \Omega(N^{\frac{1}{d}})$.

Appendix B. Upper Bound for P_d in RP2P

We establish an upper bound for P_d, $d \geq 2$, in the following. Consider an arbitrary identifier id and an arbitrary node x. Suppose x issues request(id). Each node has an equal probability of being responsible for id. $sup\_seg(x)$ consists of the segments of $(s+1)$ nodes. Hence, the probability for $id \in sup\_seg(x)$ is

$$P = (s+1)/N \tag{3}$$

It takes zero hop for the request to reach $node(id)$ if $x = node(id)$. Hence, $P_0 = 1 - \frac{1}{N}$. It takes one or less hop if id belongs to $sup\_seg(x)$. Hence, $P_1 \leq 1 - P$ =1- $(s+1)/N$. We now derive P_d for $d \geq 2$. The request will not reach $node(id)$ in d or less hops if and only if the following two conditions are satisfied.

Condition 1: $id \notin sup\_seg(x)$

Condition 2: Starting from any random neighbor of x, the request will not reach $node(id)$ in $(d-1)$ or less hops.

The probability for Condition 1 to hold is $1-P$. The probability for Condition 2 to hold is $(P_{d-1})^r$. Hence,

$$P_d = (1-P)\,(P_{d-1})^r \tag{4}$$

By induction we have, for $d \geq 2$,

$$P_d = (1-p)\sum_{i=1}^{d-1} r^{i-1} \ (P_1)^{r^{d-1}} \tag{5}$$

We simplify the formula as follows.

$$P_d < (P_1)^{r^{d-1}}$$

$$< (1-\frac{s}{N})^{r^{d-1}}$$