

# Adaptive Routing of QoS-constrained Media Streams over Scalable Overlay Topologies

Gerald Fry and Richard West  
Computer Science Department  
Boston University  
Boston, MA 02215  
{gfry,richwest}@cs.bu.edu

## Abstract

*Current research on Internet-based distributed systems emphasizes the scalability of overlay topologies for efficient search and retrieval of data items, as well as routing amongst peers. However, most existing approaches fail to address the transport of data across these logical networks in accordance with quality of service (QoS) constraints. Consequently, this paper investigates the use of scalable overlay topologies for routing real-time media streams between publishers and potentially many thousands of subscribers. Specifically, we analyze the costs of using  $k$ -ary  $n$ -cubes for QoS-constrained routing. Given a number of nodes in a distributed system, we calculate the optimal  $k$ -ary  $n$ -cube structure for minimizing the average distance between any pair of nodes. Using this structure, we describe a greedy algorithm that selects paths between nodes in accordance with the real-time delays along physical links. We show this method improves the routing latencies by as much as 40%, compared to approaches that do not consider physical link costs. Additionally, we discuss an approach that dynamically repositions nodes in logical space, to improve the likelihood of meeting service requirements on data routed between publishers and subscribers.*

## 1. Introduction

Recent work in the area of Internet-scale distributed systems suggests that a carefully constructed overlay topology is beneficial for routing application-specific data. The NARADA protocol, for instance, provides strong evidence that implementing multicast functionality at the end-host level results in advantages that outweigh the delay penalties incurred over implementation in the network core [8]. Such advantages include: (1) the ability to scale to larger topologies without requiring that group state be kept at core network routers, (2) flexibility to adapt routing behavior to application-specific events, and (3) reliance only on unicast

functionality implemented at the network layer, permitting the use of COTS-based systems on existing IP networks.

Although NARADA gives a convincing argument for the usefulness of end-system multicast routing, the protocol itself fails to scale as group sizes increase beyond a few hundred hosts, partly due to communication overheads introduced by random probe messages. In contrast, there have been efforts to generate more scalable overlays [15] for storage and retrieval as well as routing of data items among peers using consistent hashing techniques. Such work includes Pastry [17], Scribe [4], CHORD [20], CAN [16] and Tapestry [22]. There has also been work in the domain of distributed computing involving  $k$ -ary  $n$ -cube structures for communication in parallel processing architectures [9, 6, 7]. However, unlike NARADA, these systems make no explicit attempt to route data in accordance with latency and bandwidth requirements.

**Contributions:** This work focuses on the scalable delivery of real-time media streams (e.g., as part of a nationwide digital broadcast system on the scale of Shoutcast [19] but supporting live video feeds). We present an analysis of  $k$ -ary  $n$ -cube graphs as structures for overlay topologies [13]. In particular, we develop a method for determining the optimal values of  $k$  and  $n$ , to represent a logical topology supporting  $m$  physical hosts. We describe a greedy algorithm for routing over the overlay structure while taking physical network proximity measures into account. Additionally, we investigate methods for dynamic subscriber relocation in logical space based on network proximity and per-subscriber latency constraints. Simulation results show a significant reduction in delay penalties relative to unicast delays when using the greedy routing algorithm as opposed to random and ordered dimensional routing.

Sections 2 and 3 present an analysis of adaptive  $k$ -ary  $n$ -cube overlay topologies for use in QoS-aware P2P systems. In Section 4, we compare several algorithms for routing data over a  $k$ -ary  $n$ -cube topology. This is followed by Section 5 that discusses adaptive algorithms for re-assignment

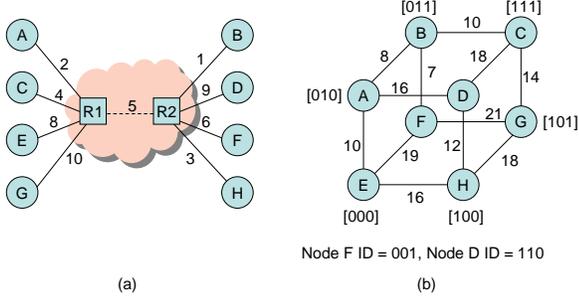


Figure 1. A sample overlay network

of hosts in logical space, in order to increase the probability of satisfying real-time latency constraints. Section 6 investigates the relationship between link stress and average latency for routed messages. Finally, related work is described in Section 7, followed by conclusions and future work in Section 8.

## 2. Analysis of $k$ -ary $n$ -cube Topologies

Scalable peer-to-peer (P2P) systems such as CHORD, CAN and Pastry use distributed hashing techniques for locating objects (and nodes) in logical space. These systems route in as little as  $O(\lg M)$  hops along the overlay topology, where  $M$  denotes the number of logical hosts communicating in the system [4, 20, 16]. Furthermore, the lookup services associated with these systems require that hosts maintain up to  $O(\lg M)$  sized routing tables.

We use undirected  $k$ -ary  $n$ -cube graphs to model logical overlays in a similar manner to the P2P systems described above. These graphs are specified using  $n$  as the *dimensionality* parameter and  $k$  as the *radix* (or *base*) in each dimension. Figure 1 shows an example of an overlay network structured as a 2-ary 3-cube graph and a corresponding underlying physical network. A cost is associated with each edge in the physical network, and each edge in the logical overlay maps to the shortest path between the respective end-point nodes in the physical topology. The costs associated with logical edges are derived as the sum of the costs along the corresponding path taken in the physical network. Note that the physical topology may contain router nodes that do not participate explicitly within the context of the overlay network (i.e., R1 and R2).

The following properties of  $k$ -ary  $n$ -cube graphs are relevant to this work:

- $M = k^n$ , where  $M$  is the number of nodes in the graph. Therefore,  $n = \lg_k M$ .
- Each node is of the same degree, with  $n$  neighbors if  $k = 2$ , or  $2n$  neighbors if  $k > 2$ .
- The minimum distance between any pair of nodes in

the graph is no more than  $n \lfloor \frac{k}{2} \rfloor$  hops.

- The average routing path length between nodes in the graph is  $A(k, n) = n \lfloor \frac{k^2}{4} \rfloor \frac{1}{k}$  hops.
- The optimal dimensionality of the graph is  $n = \ln m$ .
- Each node in the graph can be associated with a logical identifier consisting of  $n$  digits, where the  $i$ th digit (given  $1 \leq i \leq n$ ) is a base- $k$  integer representing the offset in dimension  $i$ .
- Two nodes are connected by an edge *iff* their identifiers have  $n - 1$  identical digits, except for the  $i$ th digit in both identifiers, which differ by exactly 1 modulo  $k$ .

The routing complexity in  $k$ -ary  $n$ -cubes is a logarithmic function of  $M$ . Moreover, both the average and maximum distances between nodes depend on  $k$  and  $n$ . Given the importance of these distances in QoS-constrained routing, we now show how the average hop count is derived. Note that we omit deriving the maximum hop count due to space constraints.

**Lemma 1.** For an undirected  $k$ -ary 1-cube graph, where  $k$  denotes the number of nodes in the first dimension, the sum of the distances,  $H(k, 1)$ , from any one node to every other node in the graph is given by:

$$H(k, 1) = \lfloor \frac{k^2}{4} \rfloor \quad (1)$$

*Proof.* A  $k$ -ary 1-cube can be represented as an undirected cycle of  $k$  nodes. There are two cases to consider:

- (i) If  $k$  is even,  $H(k, 1)$  is given by:

$$H(k, 1) = \sum_{i=1}^{k/2} i + \sum_{i=1}^{k/2-1} i = \frac{k^2}{4} \quad (2)$$

- (ii) If  $k$  is odd,  $H(k, 1)$  is given by:

$$H(k, 1) = \sum_{i=1}^{(k-1)/2} 2i = \frac{k^2 - 1}{4} \quad (3)$$

Thus, by Equations 2 and 3, for all  $k \geq 2$ ,  $H(k, 1) = \lfloor \frac{k^2}{4} \rfloor$ .  $\square$

**Lemma 2.** For an undirected  $k$ -ary  $n$ -cube graph, the following recursive identity holds:

$$H(k, n) = H(k, n-1) \cdot k + k^{n-1} \lfloor \frac{k^2}{4} \rfloor \quad (4)$$

Where  $H(k, n)$  denotes the sum of the distances from any one node to every other node in a  $k$ -ary  $n$ -cube graph,  $k$  denotes the radix in each dimension, and  $n$  the number of dimensions.

*Proof.* A  $k$ -ary  $n$ -cube can be divided into  $k$   $k$ -ary  $(n-1)$ -cubes. Each subcube of  $n - 1$  dimensions is connected to one or two neighboring subcubes by  $k^{n-1}$  edges. Consider a reference node in a subcube. The sum of the distances between the reference node and every other node in the same

subcube is  $H(k, n - 1)$ . The sum of distances between the reference node and each node in an adjacent subcube is  $H(k, n - 1) + k^{n-1}$ . Similarly, the sum of the distances between the reference node and each node in a subcube that is  $i$  cubed away from the subcube containing the reference node is given by  $H(k, n - 1) + i \cdot k^{n-1}$ . There are two cases to consider:

(i) if  $k$  is odd,  $H(k, n)$  is given by:

$$\begin{aligned} H(k, n) &= H(k, n - 1) + \\ &+ \sum_{i=1}^{\frac{k-1}{2}} 2(H(k, n - 1) + ik^{n-1}) \\ &= H(k, n - 1) \cdot k + k^{n-1} \frac{k^2 - 1}{4} \end{aligned} \quad (5)$$

(ii) if  $k$  is even,  $H(k, n)$  is given by:

$$\begin{aligned} H(k, n) &= H(k, n - 1) + \\ &+ \sum_{i=1}^{\frac{k-2}{2}} 2(H(k, n - 1) + ik^{n-1}) + \\ &+ H(k, n - 1) + k^{n-1} \frac{k}{2} \\ &= H(k, n - 1) \cdot k + k^{n-1} \frac{k^2}{4} \end{aligned} \quad (6)$$

By Equations 5 and 6,

$$H(k, n) = H(k, n - 1) \cdot k + k^{n-1} \lfloor \frac{k^2}{4} \rfloor. \quad \square$$

**Theorem 1.** For an undirected  $k$ -ary  $n$ -cube, where  $k \geq 2$  denotes the radix of each dimension, the sum of the distances,  $H(k, n)$ , from any one node to every other node in the graph is given by:

$$H(k, n) = k^n \cdot n \lfloor \frac{k^2}{4} \rfloor \frac{1}{k} \quad (7)$$

*Proof.* The proof proceeds by induction on the number of dimensions,  $n$ . By Lemma 1,

$$H(k, 1) = \lfloor \frac{k^2}{4} \rfloor = k^1 \cdot 1 \lfloor \frac{k^2}{4} \rfloor \frac{1}{k} \quad (8)$$

Therefore, the formula holds for  $n = 1$  and establishes a basis for the inductive argument. Next, suppose that the result holds for  $n - 1$  dimensions:

$$H(k, n - 1) = k^{n-1} \cdot (n - 1) \lfloor \frac{k^2}{4} \rfloor \frac{1}{k} \quad (9)$$

Substituting the right side of the above equation for  $H(k, n - 1)$  in Equation 4 yields:

$$\begin{aligned} H(k, n) &= (k^{n-1} \cdot (n - 1) \lfloor \frac{k^2}{4} \rfloor \frac{1}{k}) k + k^{n-1} \lfloor \frac{k^2}{4} \rfloor \\ &= k^n \cdot n \lfloor \frac{k^2}{4} \rfloor \frac{1}{k} \end{aligned} \quad (10)$$

By induction, the result holds for all  $n > 0$ .  $\square$

**Corollary 1.** In a  $k$ -ary  $n$ -cube graph, where  $n$  denotes the number of dimensions and  $k$  the radix in each dimension,

the average hop count along a path between two nodes,  $A(k, n)$ , is given by the following:

$$A(k, n) = n \lfloor \frac{k^2}{4} \rfloor \frac{1}{k} \quad (11)$$

*Proof.* The result is obtained by dividing the right side of Equation 7 by the number of nodes,  $k^n$ , and thus follows directly from Theorem 1 and the symmetry of the  $k$ -ary  $n$ -cube structure.  $\square$

The function  $A(k, n)$  is defined over the domain  $\{(k, n) | k, n \in \mathbb{Z} \wedge k \geq 2 \wedge n \geq 1\}$ , where  $\mathbb{Z}$  denotes the set of integers. Consider an extension of this function,  $A_{ext}(k, n)$ , with domain  $\{(k, n) | k, n \in \mathbb{R} \wedge k \geq 2 \wedge n \geq 1\}$ , where  $\mathbb{R}$  denotes the set of real numbers. Assuming non-integer values are possible for parameters  $k$  and  $n$ , the following optimization problem can be solved to find the optimal dimensionality,  $n_{opt}$ , with respect to the cost function  $A_{ext}(k, n)$ :

**Problem 1.** Find  $n = n_{opt}$  which minimizes  $A_{ext}(k, n) = n \frac{k^2}{4}$ , given constraints  $k \geq 2$ ,  $n \geq 1$ , and  $M = k^n$ , where  $M$  is constant.

By the constraint  $M = k^n$ , it follows that  $k = M^{\frac{1}{n}}$ . Substituting for  $k$ , the cost function becomes:

$$A_{ext}(n) = n \frac{M^{\frac{1}{n}}}{4} \quad (12)$$

Since  $M$  is held constant, taking the derivative of  $A_{ext}(n)$  with respect to the single variable  $n$  yields:

$$A_{ext}'(n) = \frac{M^{\frac{1}{n}} \cdot (n - \ln M)}{4n^2} \quad (13)$$

As a result of the constraint  $n \geq 1$ , the only relevant critical point occurs when  $n = \ln M$ . To see that this point is indeed a minimum, the second derivative with respect to  $n$  is examined:

$$A_{ext}''(n) = \frac{M^{\frac{1}{n}} (\ln M)^2}{4n^3} \quad (14)$$

For all  $n \geq 1$ ,  $A_{ext}''(n) > 0$ . In particular,  $A_{ext}''(n) > 0$  for  $n = \ln M$ . Therefore, the function  $A_{ext}(k, n)$  is minimized with respect to the constraints exactly when  $n = n_{opt} = \ln M$ , giving  $A_{ext}(k, n) = \frac{1}{4} M^{\frac{1}{\ln M}} \ln M$ . The above analysis suggests that the number of dimensions,  $n$ , of a  $k$ -ary  $n$ -cube graph with  $M$  nodes should be chosen as close as possible to  $n_{opt} = \ln M$ , in order to minimize the average path length between pairs of nodes.

### 3. $M$ -region Analysis

In practice, using  $n = \ln M$  as the number of dimensions in a  $k$ -ary  $n$ -cube graph is not feasible, since  $n$  must be a positive integer. This section introduces an integer-based analysis of parameters  $k$  and  $n$ , to construct a  $k$ -ary  $n$ -cube that is optimal with respect to average path length.

```

Calculate_M-Region(int m) {
  i = 1
  k = j = 2;
  while(M[i,j] < m)
    i++;
  n = i;

  max_M = M[i,j];
  min_A = A[i,j];

  inc_j = 1;
  while(i > 0) {
    j += inc_j;
    i--;
    if((A[i,j] <= min_A) && (M[i,j] > max_M)) {
      inc_j = 1;
      max_M = M[i,j];
      min_A = A[i,j];
      n = i;
      k = j; }
    else inc_j = 0; }
  return k, n; }

```

**Figure 2. Algorithm for calculating optimal M-regions**

Given a range of values for the number of *physical* hosts,  $m$ , in the system, a corresponding pair of values  $(k, n)$  is determined for defining an overlay network with  $M = k^n$  *logical* nodes. We refer to this range of sizes for the *physical* network,  $[m_l, m_u]$ , whose members correspond to the same chosen values of  $k$  and  $n$ , as an *M-region*. In this paper, the following assumptions are maintained for the purpose of *M-region* analysis:

- For routing to be meaningful, we require that  $m \leq M$ , with no two physical hosts mapped to the same logical node.
- The overlay structure is optimized with respect to  $A(k, n)$ , calculated to *floating point* precision.
- Given two  $k$ -ary  $n$ -cube graphs with parameters  $(k_1, n_1)$  and  $(k_2, n_2)$ , such that  $k_1 \cdot n_1 = k_2 \cdot n_2$  and  $k_1^{n_1} > k_2^{n_2}$ , the graph with parameters  $(k_1, n_1)$  is preferred. This is because it can support a larger number of *physical* hosts without increasing the average path length, given that  $A(k, n)$  is  $O(nk)$ .

In terms of this analysis, there exist some  $k$ -ary  $n$ -cube graphs that are inherently suboptimal for all values of  $m$ . For example, a 2-ary 11-cube has  $2^{11} = 2048$  nodes, while a 3-ary 7-cube has 2187 nodes. In the latter case, more physical hosts can be supported while maintaining a lower average hop count, since  $A(3, 7) < A(2, 11)$ .

**Problem 2.** Given a value for the number of physical hosts in the system,  $m$ , find  $k \in \mathbb{Z}$  and  $n \in \mathbb{Z}$  that minimize the function  $A(k, n) = O(nk)$  while simultaneously maximizing the number of nodes,  $M = k^n$ , comprising the corresponding  $k$ -ary  $n$ -cube graph.

$[m_l, m_u]$	$k$	$n$	$M = k^n$
[2, 2]	2	1	2
[3, 4]	2	2	4
[5, 9]	3	2	9
[10, 27]	3	3	27
[28, 32]	2	5	32
[33, 81]	3	4	81
[82, 243]	3	5	243
[244, 729]	3	6	729
[730, 2187]	3	7	2187
[2188, 6561]	3	8	6561
[6562, 19683]	3	9	19683
[19684, 59049]	3	10	59049
[59050, 177147]	3	11	177147
[177148, 531441]	3	12	531441
[531442, 1594323]	3	13	1594323
[1594324, 4782969]	3	14	4782969

**Figure 3. Table of M-regions**

An iterative algorithm is presented in Figure 2 for solving the above problem. The number of logical nodes is represented by  $M[n, k] = k^n$ , while the average hop distance between a pair of nodes is represented by  $A[n, k] \mid k \geq 2$  and  $n \geq 1$ .

The discussion of *M-regions* in this paper corresponds roughly to *realities* in CAN. However, the algorithm in Figure 2 is explicit in determining overlay configurations that are optimal with respect to network size and average path between pairs of *logical* hosts. The optimal  $k$ -ary  $n$ -cube graph can be computed each time the size of the *physical* network changes, or a number of *M-regions* can be pre-calculated and stored in memory, along with the state information to maintain the structures of several alternative routing topologies.

Figure 3 lists the first sixteen *M-regions* as a table with four columns, using the floating point value for average hop count,  $A[n, k] = n \lfloor \frac{k^2}{4} \rfloor \frac{1}{k}$ . This table is constructed by examining the output of the algorithm in Figure 2 for each  $m = 2^i$ , over the range  $2 \leq m \leq 4782969$ .

## 4. Proximity-based Greedy Routing

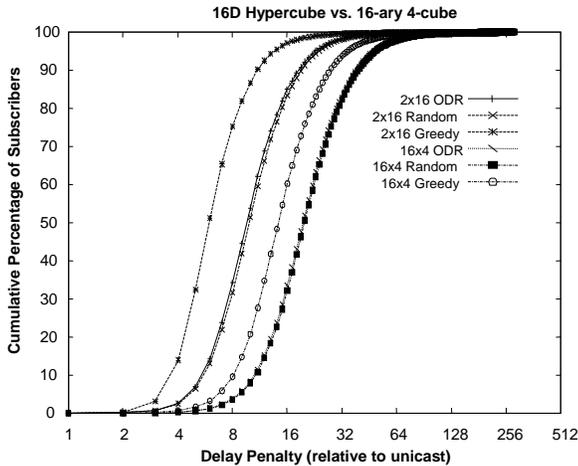
For the purposes of QoS-constrained routing, this work investigates the performance of three algorithms that leverage  $k$ -ary  $n$ -cube logical topologies, built on top of a physical network:

- *Ordered Dimensional Routing*: For a destination identifier,  $d_1 d_2 \dots d_n$ , a message is initially routed to a node that matches  $d_1$  in the first digit of its logical node ID. For each dimension  $i \mid 1 \leq i \leq n$ , the message passes to a node whose  $i$ th digit of its ID matches  $d_i$ . This is the method for routing used by systems based on Pastry, such as Scribe and PeerCQ [4, 11].
- *Random Ordering of Dimensions*: This is similar to *ordered dimensional routing* except messages are for-

warded along randomly selected dimensions towards the destination. We make sure that messages are always routed closer to the destination at each hop.

- *Greedy Routing*: As a main contribution of this work, greedy routing is performed using some measure of physical proximity. It is assumed that each host maintains a measured cost (e.g., in terms of latency) to each of its direct neighbors in the  $k$ -ary  $n$ -cube. A message is forwarded to the neighbor along the logical edge which results in the lowest cost among all other neighbors for which forwarding reduces the distance to the destination node. Since there are  $n \lfloor \frac{k^2}{4} \rfloor \frac{1}{k}$  hops on average along the overlay network between two hosts, and finding the next hop requires searching  $O(n)$  neighbors, the resulting complexity of the greedy algorithm is  $O(n^2k)$ .

**Experimental Analysis:** Given the above algorithms, we compare a number of simulations using the GT-ITM software for generating random transit-stub physical topologies [21]. The physical topology contains 5,050 routers, and the system is comprised of 65,536 hosts each randomly assigned to a router. Each experiment proceeds by choosing one host at random to be a publisher, and all other hosts are assumed to be subscribers. End-to-end delays are measured for messages routed along paths of the corresponding overlay, between the publisher and each subscriber. For comparison, unicast delays are measured for the (shortest) physical paths between the publisher and each subscriber. The relative delay penalty for each subscriber is then calculated as the logical end-to-end latency divided by the unicast delay.



**Figure 4. Comparison of routing algorithms**

Figure 4 shows the cumulative distribution of delay penalties for the three algorithms using two different configurations of  $k$  and  $n$ . The values on the  $y$ -axis represent the percentage of subscribers which incur a delay penalty

no more than the corresponding value on the  $x$ -axis. Simulation results indicate a significant improvement in delay penalty for greedy routing compared with random and ordered dimensional routing for both structures, whereas ordered dimensional routing performs no better than in the random case. We also see that the greedy algorithm performs better relative to the other routing methods when the node degree is greater, since this gives a higher probability of finding next hops with closer proximity in the underlying physical network. Additionally, the results show that the topology in which  $k = 2$  performs better than in the case where  $k = 16$ , which is consistent with the analysis of  $M$ -regions in the previous section. As can be seen from Figure 4, there is as much as a 40% reduction in the relative delay penalty when using the greedy algorithm compared to the ODR or random approaches.

## 5. Adaptive Node ID Assignment

During system initialization, it is assumed that the  $k$ -ary  $n$ -cube overlay is constructed by assigning logical node identifiers, chosen uniformly at random, to all participating physical hosts. Initially, all hosts function equally as routing agents forwarding messages across the logical topology. Once a host has received a node identifier corresponding to a position in the logical network, it can request to become a *publisher* of a new data stream or a *subscriber* to an already existing data stream. Such requests may take the form of messages routed over the optimal  $k$ -ary  $n$ -cube structure using the greedy algorithm described in the previous section.

In the absence of information about how *publisher* and *subscriber* hosts are associated with QoS-constrained data streams, random placement of physical hosts in the logical network is appropriate. This helps to minimize the average distance between nodes in an incomplete  $k$ -ary  $n$ -cube, when the number of physical hosts,  $m$ , is less than  $M = k^n$ . However, as sets of hosts begin to specify interest in receiving particular data streams with corresponding service constraints, it becomes possible to re-assign such *subscriber* hosts to more appropriate locations in logical space. Re-assignment of a host to a new location in the overlay (based on the requested service) is accomplished by *swapping* the logical node identifier, as well as routing table information, with some other host in the system.

We investigate an algorithm that swaps the positions of joining subscribers with other hosts, in order to increase the likelihood of satisfying QoS constraints. One such algorithm works as shown in Figure 5.  $S$  represents the new subscriber, which is assumed to advertise its interest in receiving a data stream from the publisher host,  $P$ . The notation  $i.cost(P)$  denotes the total end-to-end cost of routing a message between host  $P$  along the *physical* topology to host  $i$ .

```

Subscribe(Subscriber S, Publisher P, Depth d)

If d = D return

Find a neighbor i of P such that
i.cost(P) is maximal for all neighbors

If S.cost(P) < i.cost(P)
  then swap logical positions of i and S
else Subscribe(S, i, d+1)

```

**Figure 5. Adaptive node re-assignment algorithm**

The algorithm checks for positions appropriate for re-assignment of subscribers in the overlay starting from the publisher node. Each subscriber host is swapped into a position  $d$  logical hops away from the publisher host if it achieves a low enough *physical* delay to the source of the published data. Intuitively, the algorithm minimizes the maximal delay along the set of direct logical links to the publisher node by considering each subscriber host in turn. For some constant depth,  $D$ , the algorithm recursively checks for appropriate logical positions with increasing hop counts from the original publisher for relocation of the subscriber host. The algorithm is thus a branch-and-bound approach, and seeks to minimize a linear cost along a particular path in the search space. Since  $O(n)$  neighbors must be examined for each time the function is called, the resulting complexity of the adaptive algorithm is  $O(snD)$ , where  $s$  denotes the number of subscribers in the group, and  $D$  is the maximal depth from the original publisher host.

**Experimental Analysis:** Using the same simulation configurations as before, 65,536 hosts are randomly assigned to routers and then organized into a 16-dimensional logical hypercube. One host is randomly chosen to be a publisher, while a number of other hosts are randomly chosen to be subscribers. A uniformly distributed latency constraint is generated for each subscriber, in the range bounded below by the minimal physical link latency, and above by the mean physical latency multiplied by the worst case logical hop count ( $n \cdot \lfloor \frac{k}{2} \rfloor$ ). Each subscriber is then selected in random order for re-assignment to a new location in the logical overlay, using the algorithm stated in Figure 5 with  $D = 1$  and  $d$  initialized to 0. Before and after adaptation, the total cost of greedy routing over the logical topology to each subscriber host is recorded, and this cost is compared with the corresponding host’s latency constraint. A *success* is recorded if the achieved cost does not surpass the constraint for each subscriber. The total *successes* is divided by the number of subscribers in the group, to obtain a *success* ratio.

Figure 6(a) plots the success ratios for subscriber group sizes of 512, 1024, 2048, 4096, 8192, 16384, and 65535 hosts. Results are given for groups of subscribers that are

dynamically re-assigned in the overlay topology as well as for the case in which no adaptation is performed. Success ratios are consistently greater when the adaptive algorithm is used, and it is apparent that QoS constrained data streams can be more successfully delivered to subscribers when adaptive node ID assignment is leveraged.

For a given subscriber host,  $S$ , and its corresponding latency constraint,  $c$ , a *normalized lateness* value,  $L(S, c)$ , is calculated using the following formula:

$$L(S, c) = \begin{cases} 0 & \text{if } S.cost(P) \leq c \\ \frac{S.cost(P) - c}{c} & \text{if } S.cost(P) > c \end{cases},$$

where  $S.cost(P)$  denotes the total cost of routing a message along the logical network from publisher host  $P$  to subscriber  $S$ . The lateness values are normalized in order to eliminate bias towards subscribers with large latency constraints, relative to other subscriber hosts in the group, and all subscriber hosts with satisfied constraints are assigned a *normalized lateness* of zero. Figure 6(b) compares the *average normalized lateness* values obtained for varying group sizes before and after relocation of hosts in logical space. While adaptive node relocation improves performance, it is likely that better results will occur if the algorithm in Figure 5 is used with values of  $D$  greater than one.

## 6. Link Stress

The approaches considered in the preceding sections focus mainly on reducing the average total delay of routing messages between publisher and subscriber hosts. However, it is also useful to consider the effects of such algorithms on *average physical link stress*, which we define as the average number of times a message must be forwarded over each physical link, in order to multicast a distinct message from a publisher host to each of its subscribers.

An alternative algorithm, that we call “split-based greedy routing”, is used to further reduce average normalized lateness values, without unduly increasing the *physical link stress*. The algorithm extends the greedy routing approach developed in Section 4. At each hop along the path taken by the greedy algorithm, each neighbor is checked to see if it is already a subscriber. The path of a message through the *logical overlay* is redirected via an existing subscriber, if such a host exists that decreases the total end-to-end delay, compared with simply routing via the greedy algorithm.

Via simulation, we investigate the effects of the split-based greedy algorithm. A group of subscribers is formed by selecting hosts at random positions in the logical topology. For varying group sizes in a 16D hypercube topology, a single message is multicast from a randomly selected publisher to each subscriber host. Each group comprises a set of subscribers that are randomly assigned to nodes in the

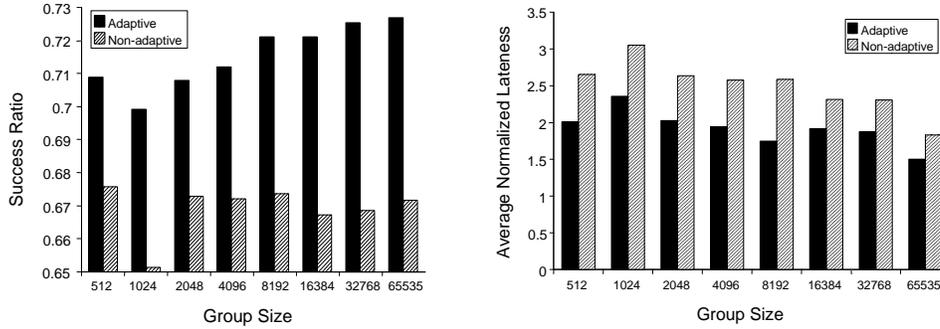


Figure 6. Effect of adaptive node relocation on (a) success ratio, and (b) normalized lateness

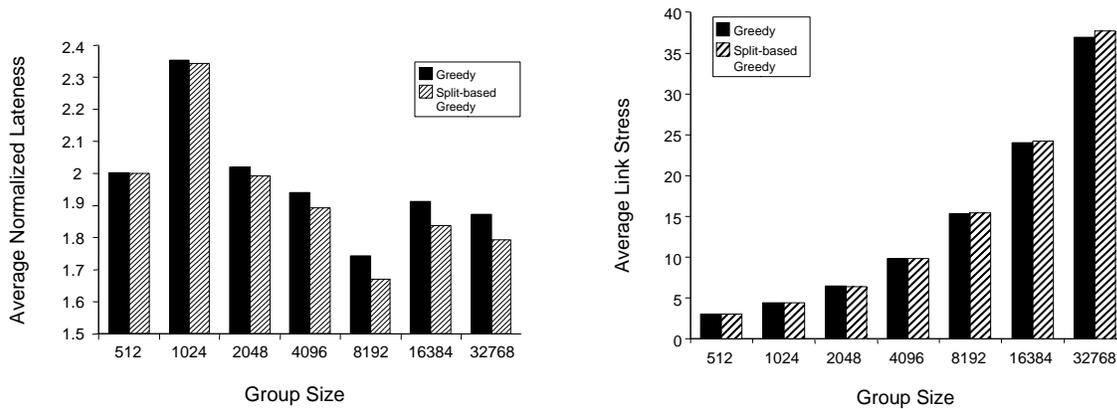


Figure 7. (a) Lateness and (b) link stress versus group size

overlay. For both of the greedy and split-based routing algorithms, a multicast tree is constructed using the *union* of paths generated by the respective routing approaches. Average normalized lateness values are calculated using the formula in Section 5, and *average physical link stress* values are obtained for each group size by counting the number of times a message is forwarded over a physical link and dividing the result by the number of unique physical links involved in multicasting the message to all subscribers.

Figure 7 shows the average normalized lateness, and average link stress, for each routing algorithm. Even though the split-based greedy approach reduces lateness, the basic greedy algorithm performs relatively well, especially in terms of link stress at large group sizes. Observe that link stress increases for larger group sizes when the mapping between *physical paths* and *logical links* results in a larger intersection of *physical links*.

## 7. Related Work

A number of systems have been developed in recent years that focus on methods for distributing data among

hosts participating in an overlay network. The taxonomy of these systems lies largely along the extremities of two dimensions: scalability and QoS awareness. For example, systems such as Pastry/Scribe, Chord, CAN, and Tapestry provide a lookup service that can scale to thousands of peers. These works differ in the formulation of their overlay topologies (e.g., Chord arranges hosts in a logical ring whereas CAN assumes a Cartesian coordinate space), but none focus on adaptive node management for QoS purposes. Approaches such as NARADA do attempt to meet QoS guarantees by considering physical proximities, but they are not particularly scalable.

While a number of other projects [1, 10, 5, 12, 2, 18, 3, 14] have implemented scalable multicast solutions at the application-level, most have either not addressed per-subscriber QoS requirements, or take a very different approach to ours. For example, OMNI [2] implements an overlay multicast network infrastructure that attempts to minimize the latency of real-time data using a two-tier approach. While facing similar objectives to our system, OMNI differs in that it divides end-hosts into two classes: (1) special Multicast Service Nodes (or MSNs) and, (2) sub-

scribing clients. MSNs form an overlay backbone and each client subscribes with a single MSN. Routing trees connecting MSNs are continuously adapted, based on network conditions and the distribution of clients. In our approach, we treat all end-hosts as equivalent peers, forming a unifying  $k$ -ary  $n$ -cube overlay for the purposes of data delivery. We select paths through this overlay based on network latencies, bandwidth availability and per-subscriber service requirements and adapt the overlay topology based on the number of end-hosts present in the system. In summary, our  $k$ -ary  $n$ -cube approach seeks to relinquish the trade-off between scalability and QoS awareness.

## 8. Conclusions and Future Work

This work analyzes the use of  $k$ -ary  $n$ -cubes for routing real-time media streams between publishers and potentially hundreds of thousands of subscribers, in keeping with per-subscriber service constraints. We analyze the minimal average hop-count between any pair of nodes in a  $k$ -ary  $n$ -cube and use this as the basis for constructing an overlay topology for real-time transport of data. Using our greedy algorithm, which leverages physical proximity information, we are able to route over such topologies with significantly lower delay penalties than existing approaches based on peer-to-peer routing.

We are in the process of building an overlay system for delivery of real-time data. Future work involves the analysis of system stability as we adapt  $M$ -regions based on host joins and departures. Likewise, efficient methods of maintaining globally consistent state to support scalable overlay networks is under investigation.

## References

- [1] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proceedings of ACM SIGCOMM*, August 2002.
- [2] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. In *IEEE INFOCOM*, San Francisco, CA, April 2003.
- [3] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth multicast in cooperative environments. In *19th ACM Symposium on Operating Systems Principles*, 2003.
- [4] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications (JSAC)*, 2002.
- [5] Y. Chawathe. *Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service*. PhD thesis, University of California, Berkeley, December 2000.
- [6] A. A. Chien. A cost and speed model for  $k$ -ary  $n$ -cube worm-hole routers. *IEEE Transactions on Parallel and Distributed Systems*, 9(2):150–162, 1998.
- [7] A. A. Chien and J. H. Kim. Planar-adaptive routing: Low-Cost adaptive networks for multiprocessors. *Journal of the ACM*, 42(1):91–123, 1995.
- [8] Y.-H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *ACM SIGMETRICS 2000*, pages 1–12, Santa Clara, CA, June 2000. ACM.
- [9] W. J. Dally. Performance analysis of  $k$ -ary  $n$ -cube interconnection networks. *IEEE Transactions on Computers*, 39(6):775–785, 1990.
- [10] P. Francis. Yoid: Extending the multicast Internet architecture, 1999. On-line documentation: <http://www.aciri.org/yoid/>.
- [11] B. Gedik and L. Liu. PeerCQ: A decentralized and self-configuring peer-to-peer information monitoring system. In *Int. Conf. on Dist. Comp. Systems (ICDCS)*, 2003.
- [12] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O’Toole. Overcast: Reliable multicasting with an overlay network. In *Proceedings of the 4th Symposium on Operating Systems Design and Implementation*, October 2000.
- [13] M. Kang, C. Yu, H. Y. Youn, B. Lee, and M. Kim. Isomorphic strategy for processor allocation in  $k$ -ary  $n$ -cube systems. *IEEE Transactions on Computers*, Vol. 52, No. 5, pages 645–657, May 2003.
- [14] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High bandwidth data dissemination using an overlay mesh. In *19th ACM Symposium on Operating Systems Principles*, October 2003.
- [15] N.J.A. Harvey, M. Jones, S. Saroiu, M. Theimer, and A. Wolman. Skipnet: A scalable overlay network with practical locality properties. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems*, March 2003.
- [16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*. ACM Press, 2001.
- [17] A. Rowstron and P. Druschel. ”Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms*, November 2001.
- [18] S. Shi and J. Turner. Routing in overlay multicast networks. In *IEEE INFOCOM*, June 2002.
- [19] Shoutcast: <http://www.shoutcast.com>.
- [20] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In *Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*. ACM Press, 2001.
- [21] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to model an internetwork. In *IEEE INFOCOM*, volume 2, pages 594–602, San Francisco, CA, March 1996.
- [22] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, Computer Science Division, U. C. Berkeley, Apr. 2001.