Dynamic Characteristics of *k-ary n-cube* Networks for Real-time Communication

Gerald Fry and Richard West

Computer Science Department
Boston University
Boston, MA 02215
{gfry,richwest}@cs.bu.edu

Abstract

Overlay topologies are now popular with many emerging peer-to-peer (P2P) systems, to efficiently locate and retrieve information. In contrast, the focus of this work is to use overlay topologies for transporting real-time streams on an Internet-scale. We aim to build an overlay communication system, for routing real-time data to potentially many thousands of subscribers with their own quality-of-service (QoS) requirements. In past work, we have analyzed the use of k-ary n-cube topologies for routing QoS-sensitive data. The basis for our work stems from the interconnection networks found in parallel communication architectures, such as the SGI Origin 2/3000. However, overlay networks addressed by our work are more scalable and dynamic than those found in traditional parallel computing platforms. We must address the issue of end-hosts joining and leaving the system at arbitrary times. While we have previously analyzed the optimal size and shape of an overlay, given the number of physical hosts in the system, this paper investigates the overheads of dynamically reconfiguring k-ary n-cubes *to deal with system membership changes. Specifically, we study the control message exchanges needed to maintain a globally-consistent snapshot of the system configuration, given dynamic joins and departures of physical hosts. A cost model, capturing the lagged response in adapting the overlay to its optimal topology is derived.*

1. Introduction

Interconnection networks in the form of *k-ary n-cube* graphs are leveraged in various parallel architectures. For example, systems such as the SGI Origin2/3000 and nCUBE n4 Streaming Media System use hypercube networks for the transfer of messages between processing elements [10]. In past work, we have shown that *k-ary n-cube* networks have desirable properties for routing messages in the context of real-time peer-to-peer (P2P) communication. Nodes are organized into an overlay topology that assumes the availability of a unicast routing service (e.g., IP). In contrast to many parallel architectures, a *k-ary n-cube* overlay may consist of many thousands of nodes with a transient membership in the system. For example, consider a nationwide digital broadcast system (on the scale of Shoutcast [14]), in which hundreds of thousands of subscriber hosts receive live video feeds from one or more publishers. It is possible to adapt the structure of the overlay to improve the performance of routing real-time data streams between publishers and subscribers. For an efficient implementation, it is thus necessary to investigate the dynamic characteristics of self-organization and adaptation in *k-ary n-cube* systems.

In prior analysis, we derive useful properties of *k-ary n-cube* graphs including formulae for average and worst case hop count between pairs of nodes, and we present an algorithm that determines the values of k and n to optimize these properties, for a given set of hosts [7]. Additionally, we study the effects of adaptation of the overlay network by re-assignment of subscriber hosts to positions closer in logical proximity to publishers of data streams. Such investigations show that adaptation of the overlay structure can result in significant reductions in routing latencies.

This work addresses the dynamic characteristics of a *k-ary n-cube* overlay network, which are related to the following aspects of the system:

- Physical host addresses are mapped to positions in the logical overlay, by assignment, and possibly re-assignment, of logical node identifiers.
- As the system evolves, the locations of publishers and subscribers change based upon application requirements and for performance reasons, such as to lower communication costs between end-systems.
- Per-subscriber QoS requirements induce adaptation of the overlay topology by re-assigning hosts to alternate positions in the logical network, in order to maintain as many service guarantees as possible.

- Joins and departures of hosts participating in the overlay configuration require self-organization mechanisms that maintain a topology that is optimal with respect to average and worst case hop count between logical nodes.

Contributions: This work analyzes the effects of dynamic joins and departures on a *k-ary n-cube* overlay topology that attempts to maintain the optimal average case and worst case hop counts between pairs of nodes, while supporting a maximal number of physical hosts. We calculate the message exchange overheads associated with join and departure events and use this analysis to derive a formula that quantifies the lag effects of bursts of such requests. Additionally, the *target M-region utilization*, or the percentage of time spent operating at an optimal overlay structure in the presence of *adaptation lag* is investigated in relation to system parameters via simulation.

2. Analysis of *k-ary n-cube* Topologies

P2P systems such as CHORD, CAN, and Pastry implicitly form toroidal topologies for locating objects (and nodes) in logical space and route messages in as little as $O(\lg M)$ hops along the overlay, where M denotes the number of logical hosts communicating in the system [3, 15, 11].

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. The following properties of *k-ary n-cube* graphs are derived in previous 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 ith 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 ith digit in both identifiers, which differ by exactly 1 modulo k.

Given a range of values for the number of *physical* hosts, m, in the system, a corresponding pair of integer values (k,n) is determined for defining an overlay network with $M = k^n$ *logical* hosts. 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*. An algorithm can be found in [7] that calculates the values of k and n to determine an *M-region* minimizing $A(k,n)$ and maximizing $M = k^n$ for a given number of *physical* hosts. Figure 1 shows the first sixteen *M-regions*.

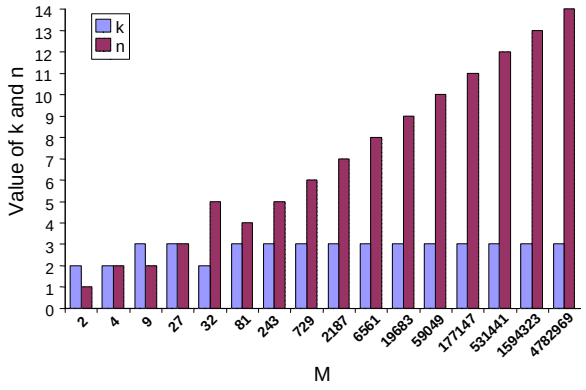

Figure 1. M-region bar-graph

3. Analysis of Dynamic Characteristics

In contrast to the preceding static analysis of *k-ary n-cube* topologies, a *dynamic* analysis focuses on transitions between M-regions due to join and departure events generated by existing and potentially participating clients. An understanding of the implications of dynamic characteristics on the overhead of maintaining the overlay structure is crucial for optimization of average hop count and handling of join/departure events to be effective.

***M-region* transitions:** For simplicity, we model an *M-region* transition as a change in the dimensionality of the *k-ary n-cube* overlay, while the radix is held constant at $k = 3$. We assume that a single *M-region* transition occurs within a given continuous time interval, (t, t'), due to the join or departure of a physical host. Let $m(t)$ denote the number of physical hosts in the system at time t. Furthermore, suppose the optimal *k-ary n-cube* topology at time t is given by the pair $(k(t), n(t))$. The *M-region* transition that occurs in the interval (t, t') is described by the following formulas:

$$k(t') = k(t) = 3$$

$$n(t') = \begin{cases} n(t)+1 & \text{if } 3^{n(t)} < m(t') \leq 3^{n(t)+1} \\ n(t)-1 & \text{if } 3^{n(t)-2} < m(t') \leq 3^{n(t)-1} \end{cases}$$

A *k-ary n-cube* graph can be viewed as a collection of k connected *k-ary (n-1)-cubes*, or *subcubes*. One such *subcube* is chosen as a *reference subcube* (i.e., the *subcube* containing all nodes that have a value of 0 for the rightmost digit in their logical identifiers). An *M-region transition* that increases the dimensionality of the topology is accomplished by *expanding* the *k-ary n-cube* graph into multiple subcubes. First, a new digit is appended to the logical identifier of each node in the graph, so that each node has an identifier consisting of $n+1$ digits. The set of nodes

spanning the original graph then becomes the reference subcube of the new topology. The $k-1$ non-reference subcubes are each constructed by copying the nodes of the reference subcube and replacing the last digit of each node's identifier in the ith non-reference subcube with the value i, $1 \leq i \leq k-1$. The resulting set of nodes are connected based on the values of their identifiers and the definition of a *k-ary n-cube* graph. Nodes within non-reference subcubes, excluding the logical position assigned to the newly joining host, become available for assignment to future arrivals and are referred to as *available logical nodes*. The resulting graph is a *k-ary (n+1)-cube*.

A *k-ary n-cube* graph can be transformed into a *k-ary (n-1)-cube* by *collapsing* all of the k subcubes into the reference subcube. That is, only one copy of the k subcubes is considered for the structure of the resulting logical network, and the last digit of each node's logical identifier is truncated, resulting in an identifier length of $n-1$ digits. Assuming physical hosts depart from the logical network at positions in non-reference subcubes, the non-reference subcubes are entirely composed of *available logical nodes* after a physical host departure causes an *M-region* transition into a smaller structure. This means that the non-reference subcubes can be safely discarded from the logical network without affecting the physical connectivity of the overlay. Note: we address the issue of departures in reference subcubes later in the paper.

Theorem 1. *Let P be the set of* assigned logical nodes, *or logical nodes that are assigned to physical hosts, and let L be the set of* available logical nodes *in a* k-ary n-cube *topology. Also, for some set of* k-ary n-cube *nodes, K, let $Span(K)$ denote the sub-graph of the* k-ary n-cube *spanned by the nodes in K. If $k(t) = 3$ is constant, and the above methods of expanding and collapsing* k-ary n-cubes *are used for successive* M-region *transitions, then the following property holds:*

$Span(L \cup P)$ *is a connected graph*

Proof. Since *M-region* transitions affect only the dimensionality of the overlay, the proof proceeds by analysis of the following two cases:

(i) Suppose a joining physical host causes the next *M-region* transition to a topology with greater dimensionality. If the property stated in Theorem 1 holds for some P with dimensionality n, then before the join occurs, $|L| = 0$ (since there would otherwise be no need for restructuring). Applying the method of expanding the reference subcube results in a topology with $n+1$ dimensions, with every node in the reference subcube a member of P and $|L| = 2 \cdot |P| - 1$. Each node in the reference subcube is connected to two distinct nodes in L, except for the reference subcube node which is connected to the joining host. Therefore, every node in L is connected to a node in P in the new topology.

logical ID	physical address
112	B
212	F
002	A
022	C
010	D
011	E

Figure 2. Sample routing table for node with logical ID 012

(ii) Suppose a host departing from the network causes the next *M-region* transition to a topology with fewer dimensions. If the property stated in Theorem1 holds for some P with dimensionality n, then immediately after the departure occurs, $|L| = 2 \cdot |P|$. Assuming that departures only affect assignment of nodes in non-reference subcubes, applying the method of collapsing the subcubes into the reference subcube results in a topology with $n-1$ dimensions, $|L| = 0$, and all *available logical nodes* in L are connected to at least one node in P.

Thus, the property stated in Theorem 1 is satisfied over all *M-region* transitions due to joining and departing of physical hosts. □

Routing State Maintenance: Each physical host participating in the *k-ary n-cube* overlay maintains a routing table with $2n$ entries, given $k > 2$. An entry consists of a logical identifier and the physical address of the host to which the logical identifier is assigned. Each entry corresponds to a direct neighbor in the logical overlay. A sample routing table is shown in Figure 2 for a host in a *3-ary 3-cube* topology. Physical addresses are shown as capital letters in the figure, although in practice the form of such addresses depends upon the underlying unicast service.

When a host joins or departs from a *k-ary n-cube* system, a number of control message exchanges are required to update the routing tables of the affected hosts. In what follows, we show the four cases that dictate the number of control messages *sent to neighbors* of a joining or departing host, as part of global routing state maintenance:

- *Join; no M-region transition*: A maximum of $2n$ messages are transmitted, since the logical node assigned to the new host may be directly connected to at most $2n$ *assigned logical nodes*. Each such neighbor is notified of the assigned logical identifier and physical address of the new host. In the best case, only one message is transmitted.
- *Departure; no M-region transition*: The worst case requires transmission of $4n$ messages, which occurs when a host departs from a logical position within the reference subcube. In this case, a host must be reas-

signed to the resulting available position from outside of the reference subcube. This requires notification of at most $2n$ logical neighbors of the removal of a physical host from a non-reference subcube. Additionally, a maximum of $2n$ neighbors are contacted with the logical identifier and physical address of the host that is reassigned to the reference subcube. In the best case only one message is transmitted.

- *Join; M-region transition*: Since a single joining host will change the dimensionality from n to $n+1$, only one physical host in the (completely occupied) reference subcube, of dimension n, needs to be informed. Hence, only one control message is needed in this case.
- *Departure; M-region transition*: $4n$ messages are transmitted in the worst case, due to a departure from the reference subcube. However, only one message is needed when a host departs from the non-reference subcube. There is no additional information that must be included in the messages beyond that which is discussed in the above case of departures, since a *k-ary (n-1)-cube* can be embedded in a *k-ary n-cube*.

The dynamic membership characteristic of *k-ary n-cube* overlay networks is modeled as a sequence of bursts of join and departure requests. Suppose a burst of size b occurs at some time $t_b \in (t, t')$, where (t, t') is a continuous time interval. The following cases are relevant to the effects of bursts on the resulting structure of the overlay topology:

- The maximum size of a burst of join requests such that no *M-region transition* is necessary is $k^{n(t')} - [k^{n(t)-1}+1]$, assuming that $m(t)$ is the smallest possible value for the corresponding *M-region* to be optimal at time t and $n(t') = n(t)$. If $m(t)$ is larger than this value, the maximum number of joins that can occur and cause no *M-region transition* is $k^{n(t')} - m(t)$.
- The upper bound on the size of a burst of departure requests such that no *M-region transition* is necessary is $k^{n(t)} - [k^{n(t')-1} + 1]$, assuming that $m(t)$ is the largest possible value for the corresponding *M-region* to be optimal at time t and $n(t') = n(t)$. If $m(t)$ is smaller than this value, the maximum number of departures that can occur and cause no *M-region transition* is $m(t) - [k^{n(t')-1} + 1]$.
- The minimum size of a burst of join requests such that exactly one *M-region transition* is necessary is $k^{n(t')} - k^{n(t)}$, assuming that $m(t)$ is the smallest possible value for the corresponding *M-region* to be optimal at time t. If $m(t)$ is larger than this value, the maximum number of joins that can occur and cause exactly one *M-region transition* is $k^{n(t')} - m(t) + 1$.
- The minimum size of a burst of departure requests such that exactly one *M-region transition* is necessary is $k^{n(t)} - k^{n(t')}$, assuming that $m(t)$ is the largest possible value for the corresponding *M-region* to be optimal at time t. If $m(t)$ is smaller than this value, the minimum number of departures that can occur and cause exactly one *M-region transition* is $m(t) - k^{n(t')}$.

Effects of Adaptation Lag: For each burst of join/departure events, there is an associated *adaptation lag*, or period of time in which control messages are exchanged to update the routing state associated with the affected nodes in the *k-ary n-cube* overlay. The *adaptation lag*, δ_b, due to a burst of join or departure events is a function of the message exchange overhead involved in servicing the individual events, and the burst size, b, that is, the quantity of join or departure requests received in the burst.

The analysis of the message exchange overhead indicates that each individual event requires $O(n)$ message exchanges to ensure global consistency of routing state. Consequently, we define the *adaptation lag* for a burst to be proportional to the dimensionality of the overlay network, n, before any *M-region* transition takes place, multiplied by the number of joins or departures associated with the burst. Hence, the *adaptation lag* is given by the following formula, where C is the *lag constant*:

$$\delta_b = n \times b \times C$$

The constant C is proportional to the average amount of time it takes to transmit a single control message containing updated routing state. *Adaptation lag* can be used to calculate the time that a *k-ary n-cube* system needs to adjust to a given *M-region transition*, in proportion to the size of the bursts that are currently being serviced. We make the distinction between a *target M-region*, or the *M-region* for which the values of k and n form a topology that minimizes average hop count for the current value of m, and an *actual M-region*, that is, a pair of values (k, n) at which the system is operating when taking *adaptation lag* into account. It is not necessarily the case that the *actual M-region* is optimal for the current value of m, if the outstanding *adaptation lag* is greater than zero at the time of transition of the *target M-region*. In other words, when discussing *actual M-regions*, we abandon the assumption that *M-region transitions*, joins, and departure events occur instantaneously, as is necessary in maintaining an optimal *k-ary n-cube* overlay over all points in time.

The value for the number of physical hosts in the system at time t is sampled at regular finite time intervals of length T. At the end of each interval, $[nT, (n+1)T]$, where $n = 0, 1, 2, ...$, the number of joins and departures that have occurred within the interval are counted and the point, $m((n+1)T)$, represents the resulting quantity of physical hosts:

$$m((n+1)T) = m(nT) + J_{n+1} - D_{n+1},$$

where J_{n+1} represents the number of join requests that occurred in the time interval $[nT, (n+1)T]$ and D_{n+1} denotes the number of departures.

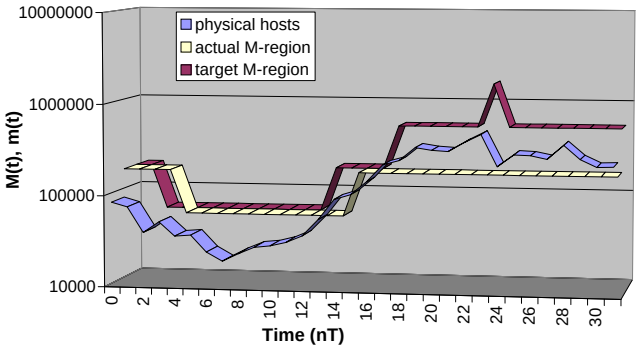

Figure 3. M-region adaptation, C = 5.0×10^{-6}

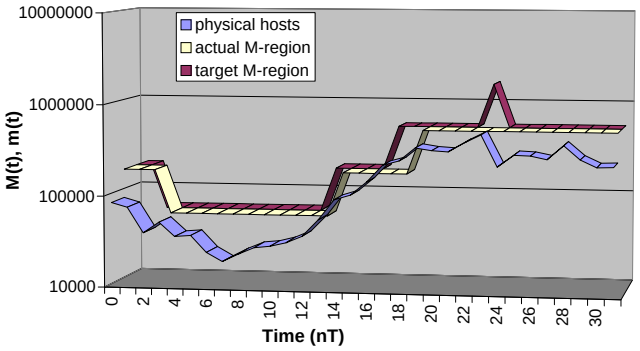

Figure 4. M-region adaptation, C = 10^{-6}

Simulation Results: A Markov-modulated Poisson process (MMPP) is used to simulate the arrivals of join and departure events [5]. The MMPP transitions between two states, that either generate bursts of join or departure requests. Values for $m(t)$ and $M(t)$ are recorded once every constant time interval, T. Furthermore, the *actual M-region* (or effective value of $M(t)$) is calculated by considering the *adaptation lag* that results from handling the burst requests. The corresponding plots of two simulations are shown in Figures 3 and 4, using different values of the lag constant, C. Each figure depicts a time delay for the changes in *actual M-region* sizes, when compared with the *target M-region* values (i.e., optimal values of $M(t)$). In each case, the delay is proportional to the change in size of m. It is also apparent from the plots that greater values for C result in a lower *target M-region utilization*, defined as the percentage of time for which the actual and target M-regions coincide. Specifically, the *target M-region utilization* is 40.6% in Figure 3, compared to 84.4% in Figure 4.

4. Implementation Issues

The implementation of the overlay system features a set of "supernodes", that are responsible for maintaining a list of logical node identifiers that can be assigned to newly joining physical hosts. That is, each host joining the system contacts one of a number of known supernodes that assigns an available logical position in the overlay network. The supernode responsible for a new host could be a default and globally-known machine, or it could be one of a set of known machines selectable by a joining host. In either case, supernodes can communicate with one another to redistribute the set of (client [1]) hosts they manage. It is desirable to have each supernode manage approximately the same number of clients, since each supernode needs to communicate with clients to reconfigure routing tables due to newly-joining and/or departing hosts.

Supernodes are notified of changes in the overlay topology through the receipt of control messages from physical hosts interested in joining or leaving the system. In response to a join or departure control message, a supernode transmits routing state updates to a number of existing clients, in order to maintain globally consistent routing state among peers. The following is a description of the types of control messages involved in managing the structure of the logical network:

- *Join message*: This type of control message is transmitted by a physical host that wishes to join the logical network. The message need only contain a header specifying its type and the physical address of the host from which it originates.
- *Departure message*: A host that wishes to depart from the overlay transmits a message containing the identifier of its position in the *logical* network and the message type.
- *Routing state update message*: Control messages of this type are transmitted by supernodes to clients in order to update routing table entries when the global routing state would otherwise become inconsistent (due to join and departure events). Routing state updates contain a routing table entry as described in the previous section. Note that an entry in an update message may contain a null value for the physical address.

Figures 5 and 6 present algorithms for handling control messages used by supernodes and end-hosts, respectively. If supernodes also participate in routing over the logical topology, then they must be equipped to handle all types of control messages. Hosts that are not designated as supernodes must only execute the algorithm shown in Figure 6.

Once a client has registered with the system, it may be used as an intermediate host for routing and data processing purposes. At any point in time, it may contact a supernode to declare a new data channel on which it wishes to publish information. Alternatively, a client may contact a supernode to obtain a list of currently available channels, to

[1] Here, a client refers to any end-host in the system, and may even include a supernode if so desired.

```
supernode_receive(message msg) {
 switch(msg.type) {
  case JOIN:
   if(M-region transition)
    add new IDs to list of available logical IDs;

   let r = random available logical ID;

   for each logical ID, i, neighboring r
    send update message to host with ID i,
    containing entry (r, msg.physical_address);

   remove r from list of available logical IDs;
   notify msg.physical_address of new ID = r;

  case DEPARTURE:
   for each logical ID, i, neighboring msg.logical_ID
    send update message to host with ID = i,
    containing entry (msg.logical_ID, null);

   add msg.logical_ID to list of available logical IDs;

  default:
   host_receive(msg);
 }
}
```

Figure 5. Response of supernodes to control messages

which it may subscribe. If a client wishes to subscribe to a channel, it sends a subscription request to the channel publisher. The overlay topology can serve as a multipurpose communication substrate, as in the combined Scribe/Pastry system [3, 12], by supporting the transportation of *control messages* as well as data streams. As a result, a subscribing client does not need to contact a supernode to register with a data channel. Instead, it routes a request to the publisher, specifying its QoS requirements on the corresponding stream.

As with Scribe, a publishing node need not be the actual source of information. This would be the case if several *real producers* of data all generated information for the same channel. In such a case, a publisher could be a *rendezvous* point for multiple data streams that need to be delivered across a single channel to subscribing nodes. Here, each producer might contact a supernode to declare interest in the same channel. Based on the physical proximities of each producer, a rendezvous point can be established at some node in the active overlay topology using a triangulation technique that minimizes the cost between each producer and the rendezvous node. This is certainly an area for further study and we intend to evaluate various approaches that are as decentralized as possible, while minimizing the amount of control state needed to be exchanged.

Additional messages exchanged between logically neighboring clients includes physical proximity information. That is, for a given overlay topology, each client exchanges with its neighbors information about the communication costs between the corresponding hosts. This information includes values such as end-to-end latency, as well

```
host_receive(message msg) {
 if (destination of msg is this host) {
  if(msg.type == ROUTING_UPDATE)
   record msg.entry in local routing table,
   overwriting any previously existing entries;
  else
   process(msg.payload); // msg is a data msg
 }
 else
  forward(msg);
}
```

Figure 6. Response of hosts to update messages

as available link bandwidth. Monitoring agents running on each host sample latency and bandwidth information at some predetermined (possibly adaptive) rate. Care must be taken to ensure the system monitors link costs fast enough to capture significant changes (e.g., when cross traffic over the Internet suddenly consumes a large amount of bandwidth and/or affects latency). Similarly, care must be taken not to sample link measurements too frequently, thereby leading to numerous redundant control messages being exchanged between hosts.

5. Related Work

A number of systems have been developed in the recent years that focus on methods for distributing data among hosts participating in an overlay network. For example, systems such as Pastry/Scribe, Chord, CAN, and Tapestry provide a lookup service that can scale to thousands of peers. However, these works are different in the formulation of their overlay topologies as well as their application. Pastry provides the functionality for routing arbitrary messages between hosts in the overlay, while CAN and Chord are systems designed for distributed storage and retrieval of data in large P2P systems. The NARADA protocol takes physical proximities into account in order to provide QoS constrained service to subscribers but fails to scale as well as those systems that leverage consistent hashing for maintaining the overlay routing topology.

While a number of other projects [6, 4, 8, 1, 13, 2, 9] have implemented scalable multicast solutions at the application-level, most have not provided a detailed analysis of the dynamic characteristics of the structure of the logical overlay, or take a very different approach to ours. Our *k-ary n-cube* topology seeks to relinquish the trade-off between scalability and QoS awareness in a highly dynamic environment. In fact, the *k-ary n-cube* system generalizes the method of consistent hashing in a graph-theoretic perspective, which is crucial for analysis of average hop count between pairs of logical nodes. The efficiency of adaptation is analyzed in terms of the cost of maintaining globally consistent rout-

ing state as bursts of join and departure requests necessitate re-organization of the overlay network.

6. Conclusions and Future Work

This work investigates dynamic characteristics of self-organizing *k-ary n-cube* overlay topologies for real-time communication between publishers and potentially hundreds of thousands of subscribers. Methods are devised for restructuring the logical network to maintain an overlay that is optimal with respect to average hop count, and algorithms are presented for keeping globally consistent routing state as the number of hosts participating in the system changes with time. We assume that the radix parameter is held constant at $k = 3$ and only the dimensionality of the overlay structure is adjusted during *M-region transitions*, by expanding the topology to include additional subcubes, or by *collapsing* the subcubes into a *reference subcube*. In analyzing this technique, we find that if all nodes in the *reference subcube* are assigned to physical hosts, then every *available logical node* represents a position in the overlay network that maintains connectivity with all other physical hosts participating in the system. This allows for a high degree of flexibility in the choice of an available logical position for assignment to hosts joining the system, and thus may have a positive impact on the percentage of per-subscriber QoS constraints that can be guaranteed on data streams.

The overhead in terms of control message exchanges associated with handling join and departure events is derived in terms of the dimensionality of the *k-ary n-cube* graph forming the overlay topology, and the upper bound on the number of message transfers required in the case of individual joins and departures is $O(n)$, whether or not the event initiates an *M-region transition*. In the case of transition to a structure with a greater dimensionality, this is accomplished by allowing some nodes to operate as if the overlay has not been restructured until it is necessary for them to participate in routing data towards direct neighbors along the additional dimension. Given the costs associated with joins and departures, the *adaptation lag* is formulated in terms of the size of burst requests, b, and the dimensionality parameter, n. If $b = O(n)$, then the *adaptation lag* is $\delta_b = O(n^2) = O(\lg^2 M)$.

We are in the process of building scalable overlay topologies in the form of *k-ary n-cube* graphs in order to analyze the performance of such systems in practice. Our aim in future work is to integrate end-host architectures for user-level sandboxing [16] with such logical networks in order to facilitate the distribution of real-time data streams. In this architecture, stream processing agents are deployed to perform application-specific computations on data at intermediate hosts while routing between publishers and subscribers.

References

[1] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. In *Proceedings of INFOCOM*, San Francisco, CA, April 2003.

[2] 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.

[3] 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.

[4] Y. Chawathe. *Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service*. PhD thesis, University of California, Berkeley, December 2000.

[5] W. Fischer and K. Meier-Hellstern. *The Markov-modulated Poisson Process (MMPP) cookbook*. Elsevier Science, 1993.

[6] P. Francis. Yoid: Extending the multicast internet architecture, 1999. http://www.aciri.org/yoid/.

[7] G. Fry and R. West. Adaptive routing of qos constrained media streams over scalable overlay topologies. In *10th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2004.

[8] 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.

[9] 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.

[10] J. Laudon and D. Lenoski. The sgi origin: A ccnuma highly scalable server. In *Proceedings of the 24th International Symposium on Computer Architecture*, pages 241–251. Silicon Graphics, Inc., June 1997.

[11] 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*, pages 161–172. ACM Press, 2001.

[12] 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 (Middleware)*, November 2001.

[13] S. Shi and J. Turner. Routing in overlay multicast networks. In *Proceedings of IEEE Infocom*, June 2002.

[14] Shoutcast: http://www.shoutcast.com.

[15] 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.

[16] R. West and J. Gloudon. User-level sandboxing: a safe and efficient mechanism for extensibility. Technical Report 2003-014, Boston University, 2003. http://www.cs.bu.edu/techreports/pdf/2003-014-user-level-sandboxing.pdf.