Cyclic Block Allocation: A New Scheme for Hierarchical Multicast Address Allocation*

M. Livingston[1], V. Lo[2], K. Windisch[3], and D. Zappala[2]

[1] Computer Science, Southern Illinois University, mliving@siue.edu
[2] Computer Science, University of Oregon, lo,zappala@cs.uoregon.edu
[3] Adv. Network Technology Ctr., University of Oregon, kurtw@antc.uoregon.edu

Abstract. This paper presents a new hierarchical multicast address allocation scheme for use in interdomain multicast. Our scheme makes use of masks that are contiguous but not prefix-based to provide significant improvements in performance. Our *Cyclic Block Allocation (CBA)* scheme shares some similarities with both Reverse Bit Expansion and *kampai*, but overcomes many shortcomings associated with these earlier techniques by exploiting techniques from the area of subcube allocation for hypercubes. Through static analysis and dynamic simulations, we show that *CBA* has the following characteristics that make it an excellent candidate for practical use in interdomain multicast protocols: better address utilization under dynamic requests and releases than other schemes; low blocking time; efficient routing tables; addresses reflect domain hierarchy; and compatibility with MASC architecture.

1 Introduction

The next decade will see increasing demands for the use of interdomain multicast as an important form of group communication to support applications ranging from multimedia transmissions to distributed conferencing and game-playing to E-commerce transactions. A critical problem that must be solved in order for multicast to serve these needs is the dynamic allocation of multicast addresses.

In this paper, we present a new hierarchical multicast address allocation scheme for use in interdomain multicast. Our scheme makes use of masks that are contiguous but not prefix-based to dramatically improve address utilization. Our *Cyclic Block Allocation (CBA)* scheme shares some similarities with both Reverse Bit Expansion [5] and *kampai* [16], but overcomes shortcomings associated with these earlier techniques by exploiting results from the area of subcube allocation for hypercubes.

We establish several fundamental results that impact the design of address allocation schemes. We demonstrate the inherent limitations of pure prefix-based address allocation with respect to its ability to recognize aggregatable blocks of

* Supported by NSF NCR-9714680

addresses. In addition, we show that the likelihood that a domain can release a block of addresses by halving is very poor. These results motivate the need for aggressive strategies such as *migration* and *swapping* to increase address space utilization and led us to the development of the *CBA* scheme.

Through static analysis and dynamic simulation we show that *CBA* has the following characteristics that make it an excellent candidate for practical use in interdomain multicast protocols: better address utilization under dynamic requests and releases than other schemes; low blocking time; efficient routing tables; addresses reflect domain hierarchy; and compatibility with MASC architecture.

2 Background

In this section, we review those approaches to address allocation that are most relevant to our approach, highlighting their strengths and weaknesses. We then briefly show the correspondence between the address allocation problem and the subcube allocation problem from parallel processing. We show how known results from the latter arena are applicable to the address allocation problem.

2.1 Terminology and notation

Throughout this paper, we use the following terminology and notation:

- block of addresses - a set of addresses that can be expressed using a *single* address expression or mask. We use the standard notation for describing a block of addresses, e.g. the set of four addresses 0000, 0001, 0010, 0011 can be represented as the *address expression* 00XX, in which the X's represent "don't care" bits. The same set of four addresses can be represented by the mask 1100. where the 1's in a mask correspond to those bit positions in an address that are significant for use in the routing table lookup. In most of this paper, we use address expressions instead of masks, but we use the terms interchangeably when the context is clear.
- prefix-based mask/address expression - one in which all the significant bits are in the leftmost positions; equivalently, in the address expression, all the "don't care" bits are in the rightmost positions.
- contiguous mask/address expression - one in which all the significant bits are contiguous (adjacent) modulo the number of bits in the address. Thus wrap-around is allowed.
- non-contiguous mask/address expression - one in which all the significants bits are located in arbitrary positions; equivalently the don't care's are also located in arbitrary positions.

For example, given a block of 2^5 addresses allocated from a 2^{10} bit address space, 00100XXXXX denotes a prefix-based address expression. 001XXXXX01 and XX00110XXX both denote contiguous address expressions, and X00XX10XX0 denotes a non-contiguous expression.

2.2 Current approaches to hierarchical address allocation

We presume a model for interdomain multicast as specified in Estrin et al. [9] and the proposals of the IETF's MALLOC working group [7]. Under this model, domains operate using the Multicast Address-Set Claim (MASC) protocol [4] for the assignment of address blocks between allocation domains. The MASC allocation domains function as nodes in the interdomain hierarchy whose composition follows customer-provider relationships between ISPs. Allocation domains claim sufficient address space from a parent domain to satisfy multicast address requests from both internal applications and child MASC domains. Internal applications are served by separate intradomain MALLOC protocols. Throughout the MASC architecture, all address allocations are granted with limited lifetimes, although renewal is possible. This allows some allocations to timeout naturally, permitting space to be reclaimed for aggregation. This avoids the alternative of forcing all applications to renumber.

Several techniques have been considered for use within the MASC/BGMP protocols for hierarchical and dynamic address allocation: *prefix-based techniques* and the *kampai* scheme which uses non-contiguous masks. Both of these techniques expand a domain's address space by doubling the size of the domain's block of addresses, and contract by halving the size of the address space. This approach is necessary to keep the routing tables small with the goal of one routing table entry per domain.

Prefix-based techniques are those that adopt the same address masking techniques as used in unicast addressing, specifying a block of addresses with a prefix-based mask. These include the scheme currently under consideration by the MALLOC working group of the IETF, *Reverse Bit Expansion (RBE)* [5], which was proposed by Estrin, Govindan, Handley, Thaler, and Radoslavov. Under the prefix-based mask, doubling occurs by changing the rightmost significant bit in the mask to a don't care bit; and halving occurs by changing the leftmost don't care bit to either a 0 or a 1. While natural and easy to understand, prefix-based address allocation is known to suffer from poor address utilization and aggregation abilities with utilization levels as low as 25% in a two-level hierarchy [5, 8].

The MALLOC proposal addresses this problem by augmenting *RBE* with the ability to migrate to a new block of addresses, at the cost of two Group Routing Information Base (*G-RIB*) entries per domain. We will demonstrate later why this extension is an excellent idea.

Tsuchiya's *kampai* scheme [16] uses non-contiguous masks with the same doubling and halving method discussed above. The use of non-contiguous masks significantly improves address utilization but suffers from several shortcomings:

- *kampai*'s bottom-up, one-address-at-a-time allocation method is too cumbersome for realistic address allocation. Obtaining a large block of addresses for a busy subdomain involves too much overhead in terms of number of requests and constant *G-RIB* updates. The variation suggested by Tsuchiya, in which addresses are allocated in chunks of 2^k-sized blocks at a time, limits address utilization.

– *kampai* allows for ambiguous or overloaded masks. This means an address mask associated with a given domain may only have part of its associated block of addresses actually used by that domain, with the remainder distributed arbitrarily to other siblings. The conflict is disambiguated in the routing tables, but results in a confusing and inelegant address scheme.
– *kampai* does not have a way to deal with address aggregation. Tsuchiya acknowledges that with *kampai* removed addresses will be scattered about the address space and that it is not possible to give back a block of address space without reassigning some addresses. Tsuchiya's simulations tested address allocation only but did not investigate a more complete model with both address requests and releases.
– *kampai*'s scheme only supports growth through doubling; it does not provide a mechanism for migration.
– *kampai*'s scheme and its data structures are hard to understand although we believe this is a problem with the presentation, not the underlying scheme.

2.3 Address allocation and subcube allocation

There is a simple and straightforward correspondence between the address allocation problem and the subcube allocation problem in hypercubes. In the former, blocks are allocated from a set of 2^n binary addresses with each block specified as a mask or address expression as defined above.

The hypercube is a recursive structure that served as the underlying communication network of the Intel iPSC and N-Cube parallel processors. In a hypercube, the 2^n processors are each labeled with an n-bit address; processors are connected in a regular pattern as illustrated in Figure 1. A subcube of size 2^k is a subset of the hypercube that itself forms a smaller hypercube. Each subcube is specified using a prefix or contiguous or non-contiguous mask in exactly the same ways as blocks of addresses are described using masks. See Figure 2 to see the relationship between subcubes and address blocks.

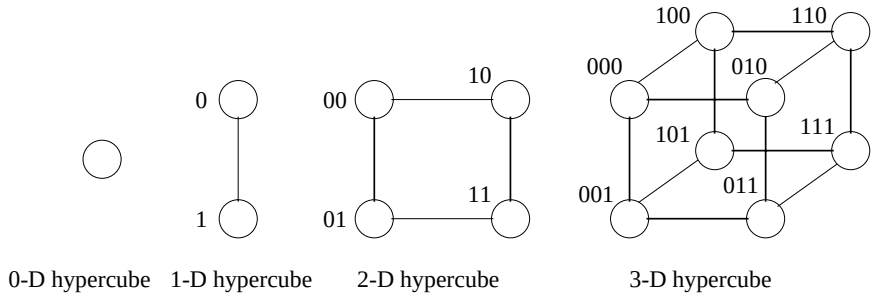

Fig. 1. Recursive definition of the hypercube

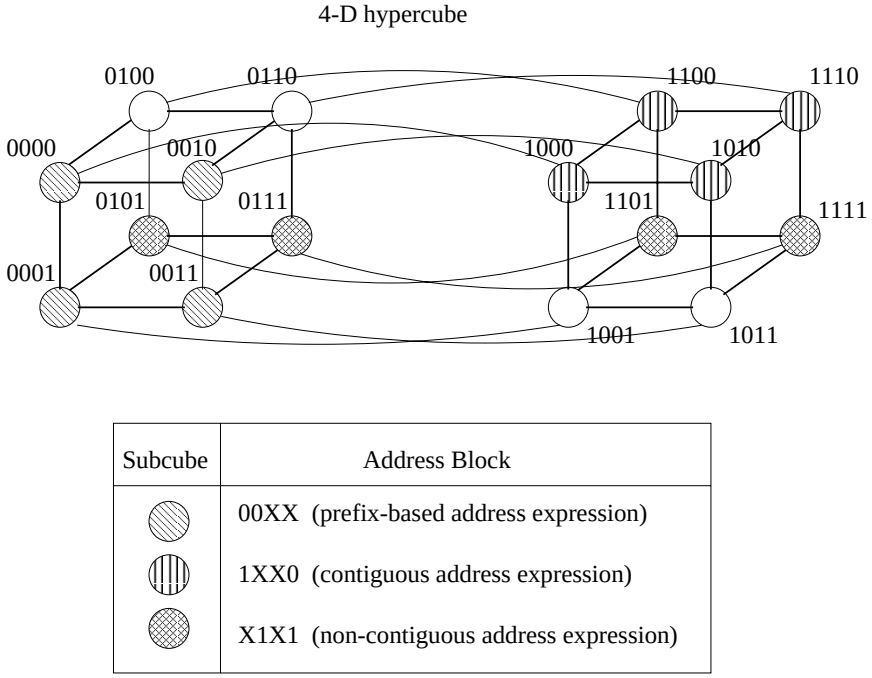

Fig. 2. The correspondence between address allocation and subcube allocation

Under address allocation, child domains request blocks of addresses, use them for a certain length of time, and then release them to the parent domain. In a hypercube machine, applications request subcubes, hold them for the runtime of the application, and then release the subcubes back to the operating system. The algorithm used by the operating system to handle the requests and relinquishments of subcubes is the *processor allocation algorithm* and has been the target of of intensive research for the past decade [14, 12, 11, 17, 3].

The key idea to remember is that a subcube is equivalent to a block of addresses that is expressible using a *single* address expression or mask. This equivalence means that subcube recognition techniques can be applied to the problem of multicast address allocation. Note, however, that while hypercubes machines typically have fewer than 2^{12} processors, the address allocation problem deals with much larger numbers of addresses: 2^{28} under IPv4 and 2^{120} under IPv6. This difference and other practical constraints associated with address allocation require that results from hypercube theory be applied to the address allocation problem with great care.

3 Principles for address allocation

From past work in the area of subcube allocation and fault tolerant hypercubes, we derive several results that have strong implications for multicast address allocation.

First, we discuss the ability of prefix-based, contiguous, and non-contiguous allocation schemes to grow by doubling the size of their address space. Next, we compare the ability of these schemes to recognize aggregatable blocks of addresses, and we show that prefix-based schemes perform poorly. Finally, we demonstrate that under any scheme, when a child domain wishes to contract by releasing one or more blocks of addresses, the likelihood of being able to contract is extremely small. Poor ability to reclaim addresses results in intolerably low address space utilization due to fragmentation of the address space.

After describing these results, we discuss their implications for the design of a practical method for hierarchical and dynamic address allocation. Most of our discussion will use the terminology from address allocation even though the original results may have been developed for hypercubes.

3.1 Doubling capability

The capacity of a scheme to expand its address space by doubling is affected by the type of mask allowed by that scheme.

In any prefix-based allocation scheme, there is only *one* choice for a new block to be combined with the current block for doubling. This can be seen by noting that doubling can only occur by converting the rightmost significant bit to a don't care bit in the address expression corresponding to the current allocation. If the single desired block needed for doubling is not free, the expansion cannot occur.

Schemes that use contiguous masks have *two* choices when expanding through doubling. This can be seen by noting that doubling occurs by converting either the leftmost or rightmost significant bit to a don't care. If neither of the two desired blocks needed for doubling is free, the expansion cannot occur.

Schemes that use non-contiguous masks have $n - k$ choices when expanding through doubling, where n is the total number of bits in the full address space, and k is the number of don't cares in the current address expression. This can be seen by noting that doubling occurs by converting any one of the significant bits to a don't care bit.

The complexity of an algorithm to double the size of a domain's block of addresses is $O(n)$ for all three cases, where n is the number of bits in the multicast address. The doubling algorithm for contiguous masks under *CBA* is described in Section 4.

3.2 Recognition capability

A given multicast address allocation scheme can be characterized by the number of distinct blocks of a given size that it is capable of recognizing. Again, the mask

pattern limits the number of blocks that are recognizable due to the respective constraints on the location of significant bits in each of these types of masks. We will see later that recognition capacity is critical for reclaiming unused addresses through migration and swapping.

Table 1 gives formulas for the recognition ability of a wide range of schemes taken from the processor allocation literature. We classify each scheme based on its use of prefix-based masks, contiguous masks, or non-contiguous masks.

The one with least recognition capability is the prefix-based *Buddy* scheme. We note that *Cyclic* with contiguous masks supports n times the recognition capability of prefix-based schemes. Also shown is the recognition capability of three schemes not considered in this paper: Gray Code and Double Gray Code [2] and Partners [1, 18]. These schemes use non-contiguous masks but do not allow for all possible patterns of non-contiguity. Interestingly, they all do worse than *Cyclic*. This is why we chose *Cyclic* for recognition in the *CBA* address allocation scheme. We originally designed *Cyclic* as a subcube allocation algorithm almost a decade ago; it is exciting to see its potential for use in multicast address allocation.

The highest recognition capability (full recognition) is associated with fully non-contiguous masks. While *kampai* uses fully non-contiguous masks, it does not support full recognition in the sense discussed in this section; *kampai* only gives an algorithm for expansion through doubling.

Figure 3 presents these recognition formulas for three sample address space sizes, 2^5, 2^{10}, and 2^{28}. The graphs show the percentage of aggregatable blocks found by a given scheme relative to the total possible number of aggregatable blocks (under full recognition). From these graphs, we see that *Cyclic* is the only scheme with reasonable performance. As the address space increases in size, all other schemes will fail to recognize blocks and *Cyclic* will only recognize large and small blocks.

It is important to note that the increasing recognition ability of these schemes comes at the cost of higher overhead. The complexity of prefix-based recognition for a given parent domain is $O(n \times C)$, where n is the number of bits in a multicast address and C is the number of child domains served by that parent. The complexity of any algorithm for full recognition with non-contiguous masks is $O(n! \times n \times C)$; the factorial term renders full recognition with non-contiguous masks computationally infeasible. The complexity of recognition for contiguous masks may be exponential in the worst case, but has not yet been proven to be so. We have found a linear time recognition algorithm for a large subclass of contiguous masks. We believe there exists a polynomial time recognition algorithm for all contiguous masks; this is part of our ongoing research.

3.3 Address aggregation capability

Our work in the area of faulty hypercubes [6, 13] yields some results that have severe implications for the likelihood of address aggregation even in the best of circumstances. The problem of reducing the set of addresses allocated to a child domain by repeatedly halving its address space is related to the problem of

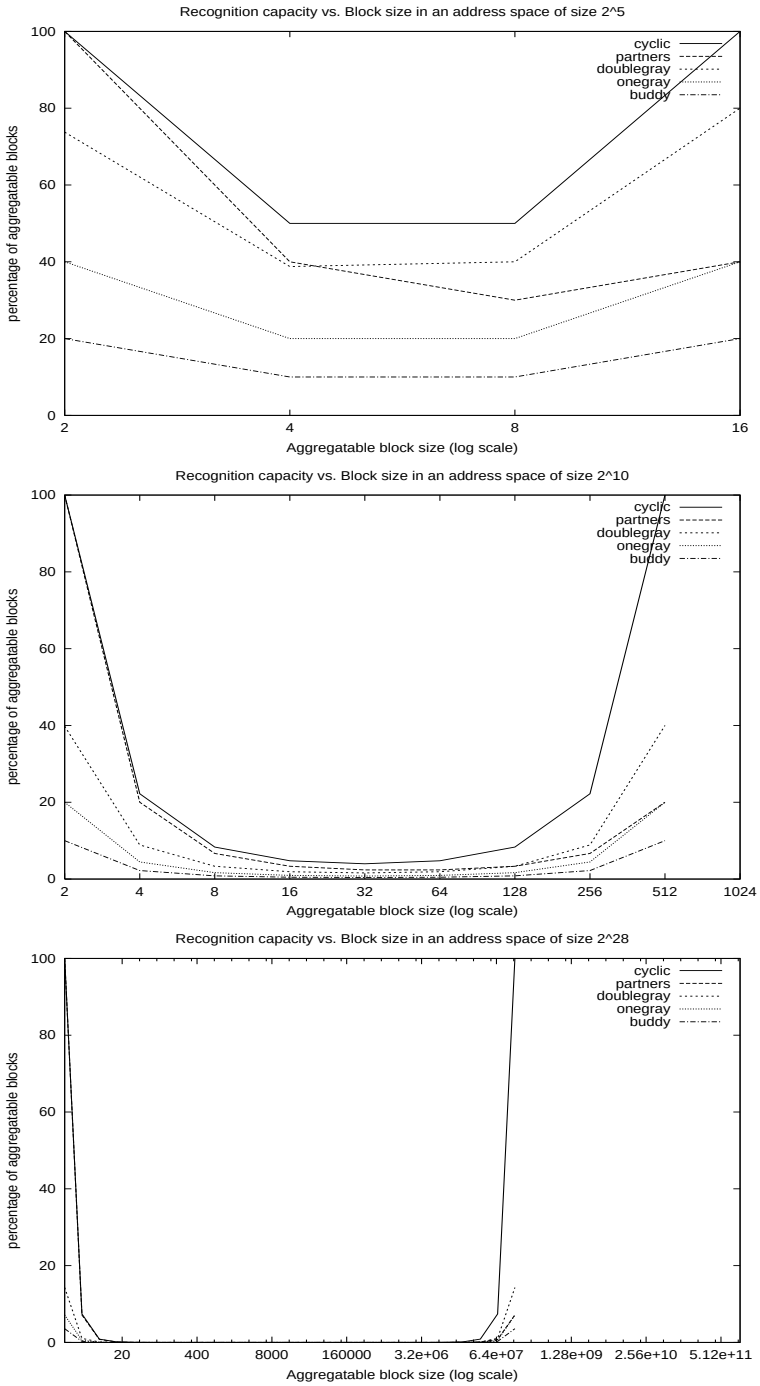

Fig. 3. Subcube recognition capacities

Subcube Allocation Scheme	Total blocks recognized	
	general formula	example: $n = 8$, $k = 3$
Buddy (prefix-based)	2^{n-k}	32
Gray (constrained non-contig)	2^{n-k+1}	64
Double Gray (constrained non-contig)	not shown	128
Partners (constrained non-contig)	$(n-k+1) \times 2^{n-k}$	192
Cyclic (contiguous)	$n \times 2^{n-k}$	256
Full (non-contig)	$\binom{n}{k} \times 2^{n-k}$	1792

Table 1. Recognition capability of prefix-based, contiguous, and non-contiguous schemes: n bit address space, k bit subcube/block

finding the minimum number of faulty processors in an n-dimensional hypercube so that every k-dimensional subcube is faulty, $k < n$. Since our focus is on contraction through halving, we first state the results for this special case. We then briefly describe the more general situation.

In particular, suppose a child domain wishes to relinquish a block of addresses to its parent by freeing half of its addresses. Specifically, if the child holds a 2^k size block of addresses it would like to free up a 2^{k-1} size block, leaving itself with a 2^{k-1} size block. However, the likelihood that 2^{k-1} or more unused addresses actually form an aggregatable block that is representable by a single mask turns out to be almost negligible.

Figure 4 shows the probabilities of being able contract by halving the size of the address space. The formula used to compute these probabilities is given in [10]. From the graph, we see that only a small number of addresses in use (around 10) from the huge exponential address space causes the probability of aggregation to fall below 5% for the samples given. The prospects become even worse for larger address spaces: an increasingly smaller fraction of the addresses in active use reduces the probability of aggregation to these low levels. In [6], we show that only two addresses in active use for multicast sessions can prevent any aggregatable block of size 2^{k-1} to exist for purposes of halving. Any pair of addresses that are complements of each other comprise such a set.

In general, our work [6] shows that with probability close to 1, a random collection of $O(n)$ faulty processors will leave no large fault free subcubes. In terms of address allocation, this means that only a small number of addresses in active use for multicast sessions can completely destroy the ability of the domain to free up a large aggregatable subset of its addresses.

We note that under the MASC architecture, this problem is diminished since rather than explicitly releasing space, MASC simply renews its claim on a smaller sub-block. The MASC protocol guarantees that any sessions allocated to the non-renewed block will have timed out by the time the sub-block times out.

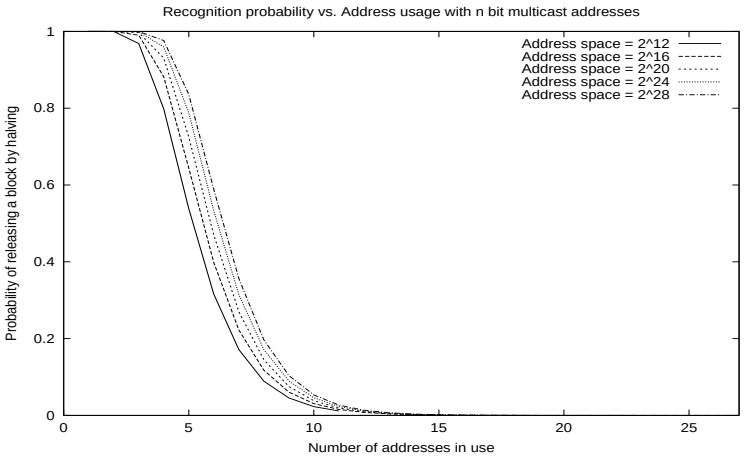

Fig. 4. Probability of address aggregation

3.4 Implications for practical address allocation schemes

Any viable address allocation scheme must perform well under dynamic expansion and contraction of its address space. Because of the woefully poor probability that a domain can actually free up a block of addresses by halving, we believe it is necessary to adopt techniques such as migration and swapping in order to aggressively reclaim fragmented blocks of free addresses.

Migration was proposed by Estrin, Govindan, Handley, Thaler, and Radoslavov in the IETF's MALLOC working group [5] in order to address the shortcomings of RBE. When a child domain needs to increase its address space and it cannot expand its current block, it migrates. Under migration, it is given a completely new block of addresses of double the size. As new multicast sessions are created, the multicast addresses are allocated from the new block. As the old sessions timeout, their addresses are freed up; when all the old addresses are free, the old block is released to the parent domain.

Swapping is a type of migration we propose in which a domain is given a new block that is the *same size* as its current block. The domain migrates to the new block and eventually frees up its old block. Swapping occurs among sibling domains, for example, when one domain can only double by using a block held by one of its siblings.

Both migration and swapping support higher address space utilization by finding alternative blocks when standard doubling and halving fails. By seeking available space within the domain's current allocation, we avoid asking the parent domain for more space for as long as possible. This is desirable since allocation of more space from the parent triggers global changes to the routing tables [9].

Migration and swapping come at a significant price: they require two routing table entries per domain during the time that the domain holds both the old

and new blocks. This becomes a severe problem for the top level domains whose current BGP routing tables contain tens of thousands of unicast entries. Under BGMP, the tables will be even further expanded to add G-RIB entries to the unicast and M-RIB entries. Thus, any scheme with fewer entries per domain is clearly preferable.

The discussion above shows a clear tradeoff among competing goals: ability to expand and contract with reasonable likelihood versus complexity of the algorithms used for expansion and contraction versus need to keep the routing tables stable and small. To summarize the analysis above:

- Independent of the type of mask, the probability of aggregation by halving is close to zero for most block sizes and especially for addresses with $n > 12$. This means migration and swapping techniques must be used to recapture fragmented free blocks.
- Migration requires good recognition capability. Prefix-based schemes have poor recognition capabilities, and full recognition (non-contiguous masks) has intolerable overhead.
- Cyclic recognition (contiguous masks) is the best known scheme that has reasonable overhead.

As a result, we designed our *Cyclic Based Allocation* scheme, which uses contiguous masks, augmented with both migration and swapping.

4 Cyclic Block Allocation (CBA)

Cyclic Block Allocation (CBA) is a new hierarchical multicast address allocation scheme that performs well under dynamic expansion and contraction of address spaces. The defining feature of the *CBA* address allocation algorithm is that it assigns blocks of addresses to child domains using *contiguous* address expressions as defined earlier. Based on our earlier discussion, the use of *contiguous* address expressions provides greater opportunities for doubling and migration than schemes that use *prefix-based* address expressions.

For ease of explanation, assume the domains are organized in a tree with a single root node and assume that the top level node initially owns the full n-bit address space. We first describe the overall operation of *CBA*. We then describe the algorithms for doubling and migration. *CBA* can be implemented as either a request-reply protocol or as a claim-collide protocol.

4.1 *CBA* description

Main Routine

- Initial request: A leaf child domain requests a base block of size 2^k. Each child domain can use a different size for its base block.
- Expansion: When a *high-threshold* is met, expand by doubling. If doubling is not possible, expand by migration. If migration is not possible, block and try again later.

- Contraction: When a *low-threshold* is met, contract by halving. If halving is not possible, contract by migration.[1] If migration is not possible, block and try again later.
- Swapping: When opportunities exist for productive swapping, a parent initiates swapping among selected siblings. (Swapping is not yet implemented.)

Subroutines

- Doubling: Doubling occurs as follows: first convert the leftmost significant bit (modulo n) to a don't care to create a new mask that represents the desired new block. This new mask is compared with the masks of all siblings to see if there is a conflict. A conflict exists between the masks for domain A and domain B if and only if the following condition holds: for every bit position in which the bits in *both* mask(A) and mask(B) are significant bits, they are identical (both one's or both zeroes). If a conflict exists, repeat with the rightmost significant bit (modulo n). If both cases fail, doubling has failed. The complexity of doubling is $O(n \times s)$, where n is the number of bits in the multicast address and s is the number of sibling domains. (This basic doubling algorithm can easily be modified for doubling with prefix-based masks and with non-contiguous masks.)
- Migration and swapping: Migration and swapping use the basic approach of the *(Full) Cyclic* subcube allocation scheme for hypercubes [13] to find an available block of addresses of the desired size; this parallel algorithm recognizes subcubes in an n-bit address space using 2^n processors. For address allocation under *CBA*, we developed a sequential algorithm for *Cyclic* to be executed on the parent node when migration or swapping is initiated. *CBA's* subcube recognition algorithm is called the *k-Cyclic* algorithm because it recognizes those subcubes whose don't care bits start in bit positions 0 through $k-1$ and which do *not* wrap around. It turns out that this limitation is needed to ensure that contiguous masks remain contiguous in a fully hierarchical address allocation scheme. This is in contrast to *Full Cyclic* which allows don't care bits to start in any position and to wrap. *k-Cyclic* runs in $O(n)$ time for fixed k. Briefly, *k-Cyclic* maintains k lists of free subcubes, one each for bit positions 0 through $k-1$. There is no duplication in recording of a free subcube from one list to another i.e., a maximal free subcube will appear only in one of the lists. The difficulty in this is the maintenance of this maximal property when a subcube is returned. We can show that for fixed k, to return a q-cube and maintain the list property will require that we investigate how the q-cube can combine with an element in one list, then further combine with an element in another list etc. One can show this process is bounded by 2^k searches; when k is fixed this number is constant. Further details about the *k-Cyclic* subcube recognition algorithm can be found in [10].

[1] Recall that under the MASC architecture halving can always be achieved without migration as discussed earlier.

4.2 CBA features

In this subsection, we describe some of the characteristics of the *CBA* scheme that make it an excellent candidate for hierarchical multicast address allocation. Results of performance evaluation experiments for address utilization, blocking time, and size of routing tables are described in the next section.

- Addresses reflect domain hierarchy: Given two or more address masks, it is easy to determine the relationship among the domains holding the corresponding blocks of addresses. In particular, domain B is a descendent of domain A *iff* the don't cares in mask(A) cover those of mask(B) and the significant bits in mask(B) cover those of mask(A). RBE's prefix-based masks satisfy this condition but *kampai* does not. Note that this property does not hold for *Full Cyclic* but does hold for *k-Cyclic*.
- Routing table size and lookup: *CBA* requires a maximum of two routing table entries per subdomain since it uses migration and swapping. Because the addresses reflect the domain hierarchy and because the masks are contiguous, the routing tables can be searched using standard search techniques. Also, because *CBA* delays requesting more address space from the parent domain through migration and swapping, fewer global changes to the routing tables are needed.
- Compatible with MASC architecture: *CBA* can function within the structure of the MASC architecture as a claim-collide protocol.

5 Simulations

The address allocation problem is highly dynamic in nature. This is particularly true when evaluating the effects of address relinquishment, which is crucial to any realistic analysis of address allocation algorithm performance. Therefore, many of the properties of the multicast address allocation schemes already presented are comparable only through the techniques of modeling and simulation.

In this section, we present a simplified model of multicast address allocation that we use to analyze the properties of *Cyclic* vs. *Prefix-based* address allocation. Our simplified model roughly approximates MASC for the purpose of investigating the fundamental principles behind these algorithms. Our ongoing work uses the detailed model and simulator proposed by Radoslavov [15] which we are using to obtain a more realistic analysis of *CBA* performance.

5.1 Algorithms Simulated

The preliminary simulations investigate the growth capability and needs for migration in the context of address block requests and releases within a single level domain hierarchy. We compare the performance of *CBA* with *Adaptive Reverse Bit Expansion* which is prefix-based. For the remainder of this paper, we refer to the latter scheme as *Prefix*.

We experimented with three versions of *CBA*, comparing their performance with two versions of *Prefix*. These differ in the method used to select a new block of addresses for migration.

- Random Fit: *CBA-RF* and *Prefix-RF* randomly choose a block from the set of free blocks of the desired size.
- First Fit: *CBA-FF* and *Prefix-FF* choose the block containing the smallest address.
- Best Fit: *CBA-BF* examines the blocks in an order corresponding to their distance from earlier allocated blocks in order to minimize potential interference[2]. We did not implement *Prefix-BF* since prior work in processor allocation has shown First Fit and Best Fit to perform comparably.

5.2 Simulation Details and Performance Metrics

We initially distribute a block of addresses to each child domain, taking the block size from a uniform distribution of block sizes up to the maximum initial block size of 20 bits in a 28-bit address space. This method was used to ramp up the simulation to a reasonable initial utilization level in order to more quickly reach steady state. Steady state was typically achieved within 5000 iterations.

A single iteration of the simulation visits all the children once in randomized order. At each visit, a child domain can initiate one of three actions: (a) a request to grow by doubling in size, (b) a relinquishment of half of the owned address space back to the parent, or (c) null action. If a request to grow cannot be satisfied by doubling in place, the failure to double is recorded and an attempt is made to migrate to a new block of double the current size. If migration cannot be achieved, the failure to migrate is recorded and the action becomes a null action. A request to halve is always satisfiable because we do not model individual session allocations or lifetimes in this simulation. When halving, the block released is selected in a manner consistent with the block selection criteria (random, first fit, best fit). The probabilities for the three actions is biased towards growth at a ratio of $5:3:2$ for grow, relinquish, null, respectively.

We modeled the load on the system by varying the number of child domains from 32 to 200, and recorded the following metrics. Results reported represent means taken over repeated runs with 90% confidence levels. The confidence intervals are not shown in the graphs below but are within $\pm 0.4\%$ of the reported means for all metrics except where noted.

Performance metrics captured by the simulation include: average address space utilization; average percentage of failed requests to double; average percentage of successful migrations; average percentage of successful growth - through either doubling or migration; average iteration number for first failure to double - this reflects the ability of the scheme to grow through doubling only; and average iteration number for first failure to migrate.

[2] This order depends on properties of the *Cyclic* allocation scheme and are discussed in [10]

5.3 Simulation Results

Below we give representative results for a 28 bit address space. Results were similar for address spaces greater than 16 bits.

Doubling (no migration): Figure 5 shows address utilization versus iteration number with migration turned off. Four schemes: *Cyclic-RF*, *k-Cyclic-RF*, *Prefix-RF*, and *Prefix-FF* are included. *k-Cyclic* has utilization at 80% and it has consistently higher utilization levels than the *Prefix-RF* scheme at 60%. Also of interest is the rate at which each scheme reaches its maximum achievable utilization level. *k-Cyclic-RF* has the fastest rate of increase. The ability to reach the plateau more quickly corresponds directly to high success rates for doubling. Note that *Prefix-FF* has very poor performance (less than 3% utilization) because the tight packing of the blocks severely limits the ability to double. In contrast, random schemes distribute the blocks, leaving empty spaces for future growth through doubling.

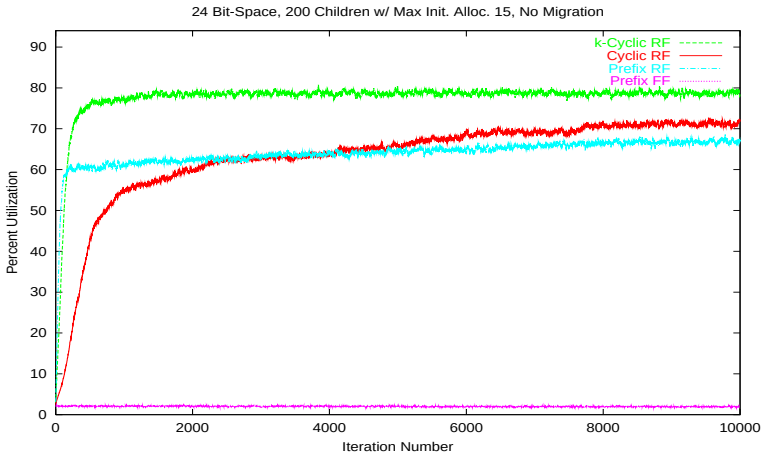

Fig. 5. Address utilization without migration (Note: legend lists algorithms in best to worst order. *Prefix-FF* is at the very bottom of the graph.)

Doubling plus migration: Figure 6 and Figure 7 illustrate the performance of the address allocation schemes when the migration feature is activated. *k-Cyclic-RF* and *Cyclic-RF* perform best, and four out of the five versions of *Cyclic* outperform the two versions of *Prefix*. Only *k-Cyclic-FF* has utilizations comparable to *Prefix*. However, the differences among all seven algorithms is only a few percentage points. What is particularly surprising is that with respect to address utilization, each individual scheme performs better without migration than with migration (except for *Prefix-FF*). We speculate that the addition of migration tends to fragment the address space in a manner that interferes with the ability

to double. We believe this fragmentation can be controlled by refinements of the migration selection criteria. Further investigation of this phenomenon is part of our ongoing work.

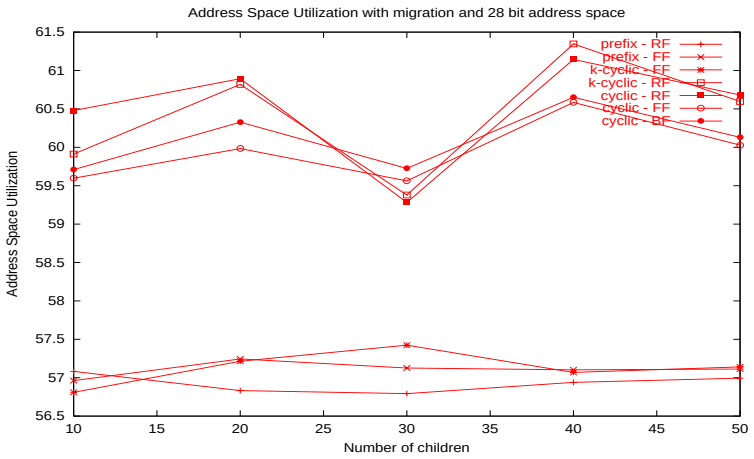

Fig. 6. Address utilization versus number of children, with migration

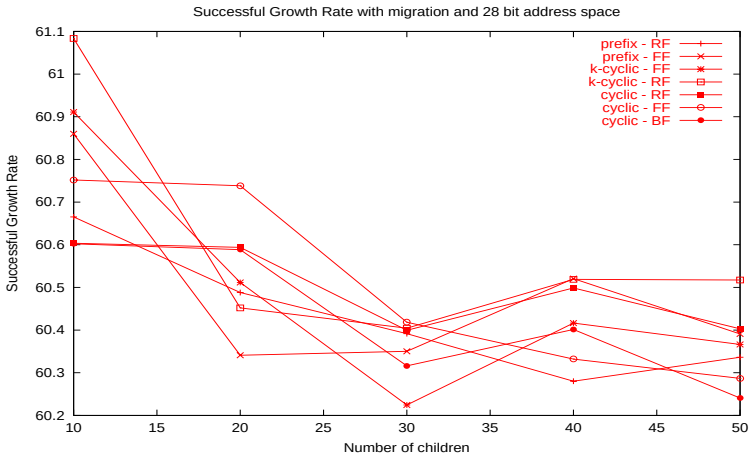

Fig. 7. Growth success rate versus number of children, with migration

Migration success rate: Figure 8 shows stark differences in the ability of these algorithms to migrate. It appears that the First Fit algorithms perform better for migration, while the Random Fit algorithms perform better for doubling. Also, while the confidence intervals for all other metrics were very small (less than 0.4%, for migration success rate the confidence intervals ranged from 0.4% to 18%, indicating that subcube recognition capacity is highly sensitive to the dynamic situation as well as the algorithm.

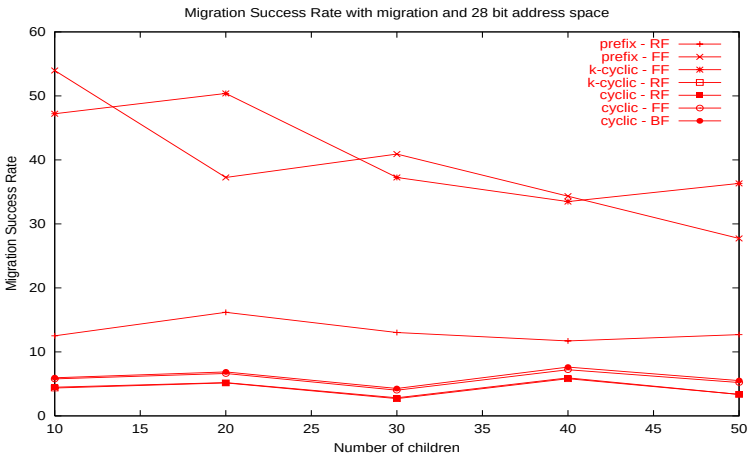

Fig. 8. Migration success rate versus number of children

5.4 Dynamic Features of *CBA*

- Address utilization and blocking time: *k-Cyclic* outperforms all other schemes and quickly reaches high levels of utilization (up to 80%) without migration. Its high success rate for growth implies less blocking time waiting for the parent domain to acquire more addresses.
- Impact on G-RIB tables: The ability to grow through doubling versus migration affects G-RIB table sizes. Growth through migration requires retaining two G-RIB entries for a period of time, a high cost situation given the current explosion in routing table sizes. Growth solely through doubling avoids this overhead. Growth success rates directly affect G-RIB flux. Whenever a growth request fails, the parent domain must also make a growth request to its parent domain, necessitating global changes to G-RIB tables at the higher level domains.

While we have run many simulations over a wide parameter space, the results are not yet conclusive. The preliminary nature of these simulations call for continued

study of *CBA* to understand the conditions under which its performance can attain its theoretical promise.

5.5 Realistic simulations

Much more detailed and realistic simulations are being conducted using the MascSim software [15], written by Pavlin Radoslavov at USC-ISI, modified to model the nonwrapping version of CBA. MascSim realistically simulates in detail the MASC protocol including hierarchical domain topologies, arbitrary address demand step functions, link failures, and true MASC claim-and-collide behavior. The most important benefit of this level of simulation will be a better analysis of the comparative G-RIB sizes and flux generated in a hierarchy by *CBA* versus the *RBE* prefix algorithm. MascSim will also provide observations of utilization and allocation latency.

6 Future Work and Conclusion

Our ongoing and future plans include refinement of our current migration algorithms with better subcube selection criteria to avoid the interference and fragmentation effect. We also plan to develop a polynomial time algorithm for *Full-Cyclic*. *CBA*'s swapping technique will be developed with new algorithms to select targets for swapping. Finally, many additional simulations are called for, most notably for migration when a child relinquishes a block and for swapping.

In general, we believe that the multicast address allocation problem is an instance of a broad class of resource allocation problems from which cross-pollination will continue to be fruitful. Much work remains to be done in both theoretical foundations and in performance analysis within the realm of hierarchical multitcast address allocation.

7 Acknowledgments

Special thanks to Prajna Dasgupta, Joannie Humphreys, and Iyer Sivaramakrishna, University of Oregon, and Josh Hoyt and Dave Meyer, Harvey Mudd College for their contributions to this project. These students were supported by NSF NCR-9714680 and an NSF Research Experiences for Undergraduates supplement to this grant. Thanks also to the referees for their helpful comments.

References

[1] A. Al-Dhelaan and B. Bose. A new strategy for processor allocation in an nCUBE multiprocessor. In *Proceedings of the International Phoenix Conference on Computers and Communication*, pages 114–118, March 1989.

[2] Ming-Syan Chen and Kang G. Shin. Processor allocation in an n-cube multiprocessor using gray codes. *IEEE Transactions on Computers*, pages 1396–1407, 1987.

[3] S. Dutt and J. P. Hayes. Subcube allocation in hypercube computers. *IEEE Transactions on Computers*, 40(3):341–352, March 1991.

[4] D. Estrin, R. Govindan, M. Handley, S. Kumar, P. Radoslavov, and D. Thaler. The multicast address-set claim (MASC) protocol. Internet Draft of the IETF MALLOC Working Group draft-ietf-malloc-masc-01.txt, February 1999.

[5] D. Estrin, R. Govindan, M. Handley, D. Thaler, and P. Radoslavov. MASC prefix allocation algorithm. Presentation given at MALLOC Working Group Meeting of IETF-41, Los Angeles, 1998. http://netweb.usc.edu/masc/ietf-41-masc-sims.ps.

[6] Niall Graham, Frank Harary, Marilynn Livingston, and Quentin F. Stout. Subcube fault-tolerance in hypercubes. *Information and Computation 102*, pages 280–314, 1993.

[7] M. Handley, D. Thaler, and D. Estrin. The internet multicast address allocation architecture. Internet Draft of the IETF MALLOC Working Group draft-ietf-malloc-arch-01, April 1999.

[8] Phillip Krueger, Ten-Hwang Lai, and Vibha A. Radiya. Processor allocation vs. job scheduling on hypercube computers. In *The 11th International Conference on Distributed Computing Systems*, pages 394–401, 1991.

[9] S. Kumar, P. Radoslavov, D. Thaler, C. Alaettinoglu, D.Estrin, and M. Handley. "The MASC/BGMP Architecture for Inter-domain Multicast Routing". In *ACM SIGCOMM*, October 1998.

[10] M. Livingston and V. Lo. A linear-time algorithm for k-cyclic subcube recognition. In preparation.

[11] M. Livingston and Q. F. Stout. Parallel allocation algorithms for hypercubes and meshes. In *Proceedings of the 4th Conference on Hypercube Concurrent Computers and Applications*, 1989.

[12] Marilynn Livingston and Quentin Stout. Distributing resources in hypercube computers. In *Proceedings of the 3rd Conference on Hypercube Concurrent Computers and Applications*, 1988.

[13] Marilynn L. Livingston and Quentin F. Stout. Fault tolerance of the cyclic buddy subcube location scheme in hypercubes. *The Sixth Distributed Memory Computing Conf. Proceedings (DMCC6)*, pages 34–41, 1991.

[14] Virginia Lo, Kurt J. Windisch, Wanqian Liu, and Bill Nitzberg. Noncontiguous processor allocation algorithms for mesh-connected multicomputers. *IEEE Transactions on Parallel and Distributed Systems*, pages 712–726, 1997.

[15] P. Radoslavov. MascSim simulator, 1999. http://catarina.usc.edu/masc.

[16] Paul F. Tsuchiya. Efficient and flexible hierarchical address allocation. *INET92*, pages 441–450, June 1992.

[17] N. F. Tzeng and G. L. Feng. Resource allocation in cube network systems based on the covering radius. *IEEE Transactions Parallel and Distributed Systems*, 7(4):328–342, April 1996.

[18] Kurt Windisch, Virginia Lo, and Bella Bose. Contiguous and non-contiguous processor allocation algorithms for k-ary n-cubes. In *Proceedings of the 1995 International Conference on Parallel Processing*, pages II–164–II–168, 1995.