Group Aggregation for Scalable Anycast Routing

Zhibin Mai, Shengquan Wang, Dong Xuan, and Wei Zhao *

Abstract

In this paper, we address the issues related to aggregation of anycast groups for scalable anycast routing. Anycast is a new network service that allows a sender to access anyone in a group that shares the same anycast address. Anycast has numerous potential applications. However, it also introduces new issues. One of the issues is that the size of the anycast routing tables is significantly increased. The main goals of this study are to reduce the size of routing tables and to improve network end-to-end performance. Particularly, we propose three group-aggregation algorithms with different objectives: (1) group-aggregation for minimizing the routing table size, (2) group-aggregation for balancing interface load, and (3) integrated group-aggregation. These algorithms take full advantages of the anycast semantics. Our evaluation results show that our group-aggregation algorithms can efficiently reduce the size of routing tables by 90% while achieving high network performance in terms of the average end-to-end delay.

1 Introduction

In this paper, we address the issues related to aggregation of anycast groups in order to improve network performance for anycast service. Being proposed in IPv6 [5], anycast is a new but important service that allows a sender to access anyone in a group that shares the same anycast address. In this context, anycast can provide service transparency for applications (e.g., DNS and mirrored web servers). It can also improves network performance by effectively sharing the loads among the members of an anycast group.

However, potentially excessive growth of routing tables in anycast routing may limit scalability of the service. The size of the routing tables significantly affects the speed of routing. As the number of end users is growing exponentially in the current Internet, the routing tables have become larger and larger. It has become a bottleneck for increasing packet forwarding. For unicast, by carefully organizing the distribution of the unicast addresses in hierachichical manner, the routing table size is reduced via address aggregation strategy, e.g., CIDR [2]. However multicast and anycast defy this form of hierarchical aggregation. A multicast or anycast address represents a group of nodes that share particular characteristics and exist somewhere in the Internet. There is no reason to require multicast and anycast group topology to be hierarchical or to comply with the unicast address topology. In [6], group-based aggregation for multicast is proposed. With this technique, the size of routing tables can be reduced dramatically. To the best of our knowledge, there has been no previous work on using group aggregation techniques for anycast.

In this paper, we study the aggregation of anycast groups in an attempt to reduce the size of routing tables and improve network end-to-end performance. Our approaches take full advantage of anycast semantics. Specifically, we consider three algorithms: (1) Group-aggregation for minimizing the size of routing tables; (2) Group-aggregation for balancing load on the interfaces; (3) Integrated group-aggregation. We conducted extensive performance evaluations on the proposed algorithms. The evaluation results show that our algorithms can reduce the routing table size by 90 percent while achieving high network performance in terms of the average end-to-end delay.

The remainder of this paper is organized as follows: In Section 2, we describe the implementation model. Three group aggregation algorithms are proposed in Section 3, and evaluation results are presented in Section 4. A summary of this paper is given in Section 5.

2 Model and Definitions

2.1 Structures of Routing Tables

The sematics of anycast service should be considered on design of the routing table structure for anycast routing. The notion of anycast presumes that multiple members possibly exist in an anycast group, which means multiple paths can be used to improve network performance. To realize multiple path routing, multiple path information would be gen-

*Zhibin Mai, Shengquan Wang, and Wei Zhao are with the Department of Computer Science, Texas A&M University, College Station, TX 77843. Email: {zbmai, swang, zhao}@cs.tamu.edu. Dong Xuan is with the Department of Computer and Information Science, The Ohio State University, Columbus, OH 43210. Email: xuan@cis.ohio-state.edu.

erated and kept in routing tables. The semantics of anycast specifies that only one member of an anycast group should receive the anycast packet, which requires *exclusive packet forwarding* among the outbound interfaces of a router.

Before proceeding further, let us review several router architectures. Generally speaking, there are two types of router architectures: one has a single forwarding engine, and another has multiple forwarding engines. For the router having a single forwarding engine, the inbound interface of the router sends the packet header to the forwarding engine, upon receiving a packet. The forwarding engine determine an outbound interface based on a central routing table. Then the inbound interface forwards the packet to the corresponding outbound interface according to instructions from the forwarding engine. Router technology is moving from architecture with single forwarding engine towards one with multiple forwarding engines. Each interface may have one forwarding engine. The overhead in making routing decisions can be distributed at all the interfaces. For the router with this architecture, upon receiving a packet, the inbound interface broadcasts it to all the outbound interfaces after it conducts a simple pre-processing. Then each outbound interface individually decides whether it should forward the packet based on its routing table. According to [3], this method yields a good performance.

In this paper, we would like to address the problems on group aggregation for anycast routing with this kind of architecture and to propose some efficient approaches. For a router with distributed routing tables, different components of the router must coordinate to select paths. To achieve this, our basic idea is to use synchronization IDs to coordinate individual forwarding engines. Our proposed structures of routing tables (see Table 1) are as follows:

- At an inbound interface, the routing table keeps one entry for each active anycast group. [1] The entry consists of two fields: one is the anycast group address and the other is the number of available paths to this group.
- At an outbound interface, the routing table keeps one entry for each active anycast group if there is one or more paths to this group passing through this interface. The entry consists of two fields: one is the anycast group address and the other is the synchronization ID. To each anycast group, the synchronization IDs of its entries at different outbound interfaces should be different.

We now discuss the correctness with this kind of routing table structure. Upon receiving an anycast packet, the inbound interface searches its routing table to determine the number of available paths, say p. Then it broadcasts the packet and an number, say q [2], $q \leq p$. Each outbound interface searches its routing table to locate the corresponding entry and compares its synchronization ID with q. If matching, the outbound interface will forward the packet; otherwise, it will ignore the packet. Since the synchronization IDs to the same anycast group at the outbound interfaces are unique, only one copy of the packet will be sent out.

group ID	# of paths
1	2
4	2
...	...
100000	1

group ID	synch. ID
1	2
4	1
...	...
100000	1

Table 1. Routing Tables at Inbound(left) and Outbound(right) Interfaces

2.2 Group-Interface Matrix and Synchronization Matrix

The following matrix describes the distribution of multiple paths of anycast groups passing through the outbound interfaces in a router.

Definition 1 (Group-Interface Matrix) *Given a router with n interfaces, there are totally m anycast groups which have paths passing through these interfaces. We define a $\{0,1\}$-matrix $A = (a_{ij})_{m \times n}$ as the group-interface matrix for this router, where the row index represents group ID [3], the column index represents interface ID. Each entry a_{ij} in the matrix is defined as below:*

$$a_{ij} = \begin{cases} 0, & \textit{group } i \textit{ has no path through interf. } j \\ 1, & \textit{group } i \textit{ has path through interface } j \end{cases} . \quad (1)$$

An example of group-interface matrix A of a router with 4 interfaces and 5 anycast groups, is shown as follows:

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{pmatrix} . \quad (2)$$

We use a concept of block to represent the successive non-zero entries in the same column within the interface matrix.

Definition 2 (Block) *For matrix $A = (a_{ij})_{m \times n}$, we define a series of successive entries with value 1 in a column j from row i_1 to row $i_2 - 1$ as a block $B[i_1, i_2; j]$, where $i_2 - i_1$ is defined as its length.*

[1] By an active anycast group of a router, we mean the router can receive packets targeting this anycast group.

[2] q can be determined by many approaches, e.g., random integer.

[3] In the following, if the context is clear, we use the row index and the group ID interchangeably.

The routing tables at all outbound interfaces in a router are combined to form a p-path synchronization matrix.[4]

Definition 3 (Synchronization Matrix) *Given a group-interface matrix $A = (a_{ij})_{m\times n}$, we define matrix $A^p = (a^p_{ij})_{m\times n}$ as a p-path synchronization matrix, where all entries a^p_{ij}'s satisfy: i) $a^p_{ij} = 0$ in A^p if $a_{ij} = 0$ in A; ii) For each group i, there are n_i non-zero entries a^p_{ij}'s which are synchronization IDs $1, 2, \ldots, n_i$ respectively, where $n_i = \min\{p, \sum_{j=1}^{n} a_{ij}\}$.*

Given synchronization matrix, successive entries with same synchronization ID to same interface form a *range*.

Definition 4 (Range) *For p-path synchronization matrix A^p, we define a set of maximal [5] successive entries with synchronization ID k in interface j from group i_1 to group $i_2 - 1$ as a range $R^k[i_1, i_2; j]$, where $i_2 - i_1$ is defined as its length. We define $|A^p|$ — the size of the p-path synchronization matrix A^p — as the number of all ranges in A^p.*

Given group-interface matrix A, for a p-path synchronization matrix A^p, we define the *compression ratio* of A^p over A as $CR = \frac{|A^p|}{m*n}$. For each interface j, we define its *load* L_j as the number of assigned synchronization IDs (non-zero entries in A^p) for interface j. The *compression ratio* and *load* are two main metrics to measure the efficiency of group aggregation algorithms.

3 Group Aggregation

3.1 Overview

The main goals of this study are to reduce the size of routing tables and to improve end-to-end network performance. A novel "group-aggregation" approach is proposed to achieve our goals. Its basic idea is to combine multiple entries in the routing table into one entry. According to Subsection 2.1, there are two main fields in an entry: the first one is group ID; the second one is the number of available paths to the routing table at the inbound interface and the synchronization ID to the routing table at the outbound interface. For multiple entries to be combined in one, the values of the first fields(i.e., group IDs) should be successive and the values of the second fields should be the same.

In our group-aggregation approach, in order to efficiently aggregate the routing table, we do take into account the following special features: i) *Not all anycast groups are active.* Usually, many anycast groups are not active and there is no entry for them in the routing table. This may result in inconsecutiveness of group IDs in a routing table. By inserting some entries of these inactive groups, the group IDs may become consecutive and the relevant entries can be combined. This idea was first proposed in [6] for multicast. It is certainly applicable to anycast too. With this feature, we can treat all group IDs as successive ones. ii) *Not all paths need to be utilized.* The semantics of anycast require routing any anycast packet to one of the anycast group members. For a given router, it is unnecessary to use all the paths of an anycast group (as long as the paths are more enough for load balance). In a routing table, we can remove the entries that represent the paths, which the router does not need. By effectively removing such entries, we may be able to further reduce the size of the routing table.

With the above ideas in mind, we discuss our group-aggregation approach. Given the number of paths required by multi-path routing, the number of available paths for each group is determined. Then the routing table at the inbound interface is fixed and its group aggregation is straightforward. However, that is not the case for the routing table at the outbound interface. Synchronization IDs must be assigned. It is a challenge to guarantee exclusive forwarding and multi-path routing. In the following sections, we will focus on the group aggregation for routing table at the outbound interface. [6]

Exclusive forwarding and *multi-path routing* make all entries of the routing tables for any specific group relate to each other. *Synchronization matrix* fully discloses this kind of relationship. It combines the routing tables at all interfaces. In synchronization matrix for each group, non-zero synchronization IDs are different for exclusive forwarding and there are totally p non-zero synchronization IDs for p-path routing. On the other hand, synchronization matrix is based on *group-interface matrix*.

By using these two matrices, our group-aggregation approach can be described more specifically: given group-interface matrix $A = (a_{ij})_{m\times n}$ with p-path routing requirement, the approach needs to carry out two tasks: i) *Interface determination*: As required by p-path routing, the routing table will keep p interfaces, if available, for each group. This task focuses on selecting p interfaces in all potential interfaces for each group.ii) *Synchronization ID Assignment*: To achieve exclusive routing for any specific group, different synchronization IDs need to be assigned.

Once these two tasks are finished, a synchronization matrix A^p will be obtained. For example, with different interface decision and synchronization ID assignment on group-interface matrix A in (2), three synchronization matrices

[4] "p-path (routing)" describes such a multi-path routing strategy: A router, having more than p available paths to an anycast group, will only pick p paths. Otherwise, it will select all available paths.

[5] For $R^k[i_1, i_2; j]$, "maximal" means that $a_{i_1-1,j} \neq k$ and $a_{i_2,j} \neq k$.

[6] In the rest, the routing table is the one at the outbound interface.

A_x^3, A_y^3 and A_z^3 are obtained as listed in (3).

$$\begin{pmatrix} 2 & 1 & 3 & 0 \\ 0 & 1 & 0 & 0 \\ 2 & 1 & 3 & 0 \\ 2 & 1 & 0 & 0 \\ 2 & 1 & 3 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 2 & 3 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \\ 2 & 1 & 0 & 0 \\ 2 & 0 & 1 & 3 \end{pmatrix} \begin{pmatrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 0 & 0 \\ 2 & 1 & 0 & 3 \\ 2 & 1 & 0 & 0 \\ 2 & 0 & 1 & 3 \end{pmatrix} \quad (3)$$

Recall that in a synchronization matrix, one column corresponds to the routing table at one interface. One row represents the distribution of paths to one anycast group among all interfaces. For these three synchronization matrices, we can obtain their compression ratios, and load distribution for all interfaces based on the definitions given in Sec. 2.2. For example, A^x has 6 ranges, so its size is 6. There are 5 interfaces and 4 anycast groups. Hence the compression ratio of A^x is $\frac{6}{5*4} = 0.3$. Recall that the load on the outbound interface is the number of non-zero entries in the matrix, hence the load on interface 1,2,3 and 4 are 4, 5, 3 and 0 respectively. The results are shown in Table2.

synch. matrix	compression ratio	load			
		Intf 1	Intf 2	Intf 3	Intf 4
A_x^3	0.30	4	5	3	0
A_y^3	0.45	3	3	3	3
A_z^3	0.35	3	4	2	3

Table 2. Three synchronization matrices

The above example shows that different interface decision and synchronization ID assignment will lead to a different compression ratio and load; hence, this will lead to different size of routing tables and network performance. In the following, we present three group aggregation algorithms to achieve different objectives: i) Group aggregation for minimizing the routing table size: this algorithm aims to achieve the minimum size of routing tables. ii) Group aggregation for balancing interface load: this algorithm aims to achieve high network performance through balancing load among the outbound interfaces. iii) Integrated group aggregation: this algorithm integrates the above two algorithms to minimize the routing table size and to achieve high network performance.

3.2 Group Aggregation for Minimizing the Routing Table Size

The objective of this algorithm is purely to reduce the size of routing tables. To differ it from the other two algorithms, we call it *greedy aggregation* algorithm in short.

3.2.1 Algorithm

This algorithm carries out synchronization ID assignment first, then implicitly performs the task of interface determination. The whole algorithm (see Figure 1)is a loop apart from some initial work [7]. In one session of the loop, there are two key steps: (1) Determining the synchronization ID that will be assigned to the block chosen later; (2) Choosing the longest block and assigning it the determined synchronization ID, to form a range.

Input: group-interface matrix A.
Output: p-path synchronization matrix A^p.
1. initialize A^p with all entries equal to zero;
2. initialize depth $D^k = 1$ for range subcollection RC^k, $k = 1, 2, \ldots, p$;
3. do
  3.1. find the minimum depth, *i.e.*, $\min\{D^k\}$, and set k^* as the minimum of synchronization IDs that achieve the above minimum depth;
  3.2. $i^* = D^{k^*}$; /* Next starting point(group ID) for RC^{k^*} */
  3.3. if $n_{i^*} < k^*$ /* $n_{i^*} = \min\{p, \sum_{j=1}^{n} a_{i^*j}\}$ */
    3.3.1. $D^{k^*} = D^{k^*} + 1$;
    3.3.2. goto step 3.1;
  else
    3.3.3. choose the longest unassigned block $B[i^*, i^{**}; j^*]$ with $n_i \geq k^*$ for all $i \in [i^*, i^{**} - 1]$ and assign it synch. ID k^* to form range $R^{k^*}[i^*, i^{**}; j^*]$ into A^p;
    3.3.4. update depth D^{k^*} for range subcollection RC^{k^*};
  while $(D^{k^*} \leq m)$
4. return A^p.

Figure 1. Greedy Aggregation

In this algorithm, the interface determination is implicitly conducted once the synchronization ID assignment is done. The final synchronization matrix A^p would meet the requirement of p-path routing because of the following reasons: (1) Initially, all entries in the synchronization table A^p, are set to zero; (2) During the process of synchronization ID assignment, only the entries that are assigned synchronization IDs are inserted into A^p; and (3) The assigned synchronization ID is not larger than p. Figure 2 shows how the algorithm works on group-interface matrix A in (2) to achieve 3-path routing.

3.2.2 Discussion

In terms of the number of non-zero entries for each group in the group-interface matrix to do p-path routing, there are two different cases:

Case 1: The number of non-zero entries for each group is not less than p: Specifically, The related group-interface matrix $A = (a_{ij})_{m \times n}$ meets the following condition:

$$\sum_{j=1}^{n} a_{ij} \geq p \quad (4)$$

[7] A concept, called depth D^k, is used to describe the algorithm. D^k is defi ned as the maximum end point i_2 for all chosen range $R^k[i_1, i_2; j]$'s in range subcollection RC^k associated with synchronization ID k.

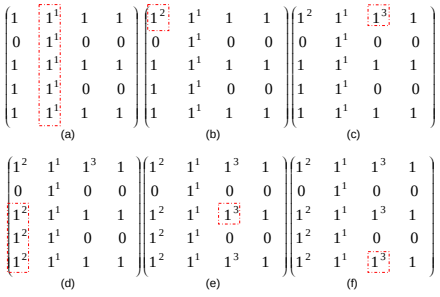

Figure 2. An example of *greedy aggregation* process: In steps (a) – (f), all supercase numbers represent synch. IDs and each square includes the chosen unsigned longest block.

holds for $i = 1, 2, \ldots, m$.

In this case, we present an important theorem below. The proof of this theorem is given in Appendix A.

Theorem 1 *Given group-interface matrix $A = (a_{ij})_{m \times n}$, for p-path routing, if (4) holds, the algorithm of group-aggregation for reducing the routing table size (or the synchronization algorithm) can minimize the routing table size, i.e. can achieve the optimal compression ratio on group-interface matrix A.*

Case2: The number of non-zero entries for some group is less than p: More specifically, For the related group-interface matrix $A = (a_{ij})_{m \times n}$, (4) does not always hold for all groups $i = 1, 2, \ldots, m$.

In this case, we find that our algorithm does not always achieve optimality. However, a lower bound for the size of the obtained synchronization matrix can be derived, which can be used to evaluate the performance of our algorithm. The following theorem gives this kind of a lower bound.

Theorem 2 *Given group-interface matrix $A = (a_{ij})_{m \times n}$, under any group-aggregation algorithm, the size of synchronization matrix A^p is bounded by Δ, i.e. $|A^p| \geq \Delta$, where Δ is a constant depending on A. Therefore the compression ratio RC on the routing table by the greedy aggregation algorithm can be bounded, i.e. $RC \geq \frac{\Delta}{m*n}$.*

The theorem states that no matter how good the group-aggregation algorithm is, the size of its resultant synchronization matrix is not less than a value which only depends on the group-interface matrix. Our evaluation data shows that the compression ratio achieved by our *greedy aggregation* algorithm is close to this lower bound, *i.e.*, $RC \sim \frac{\Delta}{m*n}$. Hence, the *greedy aggregation* algorithm can approximately achieve optimal compression ratio.

A method to determine the lower bound Δ and the proof of this theorem are given in Appendix B. In the proof, we see that *Theorem 2* also holds for case 1, and $\Delta = |A^p|$, thereby $RC = \frac{\Delta}{m*n}$.

3.3 Group Aggregation for Balancing Interface Load

Load-balancing aggregation [8] is an extension of the greedy aggregation by balancing load among the different interfaces. Note that the greedy aggregation algorithm conducts the task of synchronization ID assignment first, and the task of interface determination, i.e. resetting some non-zero entry to achieve p-routing routing, is performed implicitly. In our load-balancing aggregation algorithm, interface determination will be conducted first. The non-zero entries in the group-interface matrix will be selected to reset with the purpose of load balance, while meeting the requirement of p-path routing.

3.4 Integrated Group Aggregation

Integrated Group Aggregation algorithm aims to enhance the load balance algorithm while adding an extra-phase to achieve high compression ratio in interface determination. It does a trade-off between load balance and compression ratio. In this algorithm, entries in some short blocks are selected to be reset in the newly added phase. Recall that in the load balancing algorithm, entries in the interface-matrix will be selected to be reset for the purpose of load balance until the weights of all groups are not larger than p. Hence, with entries in some short blocks being reset, the load balancing algorithm has less freedom to reset other entries. Accordingly, entries in the long blocks will have less opportunities to be reset, and can be formed into ranges in the *Greedy Aggregation* algorithm, which is called by the *load-balance aggregation* algorithm. With the *greedy aggregation* algorithm, we state that selecting long blocks to form ranges can increase compression ratio. We expect this algorithm to achieve much higher compression ratio than the one in the load-balance algorithm.

4 Performance Evaluation

In this section, we evaluate the performance of our proposed group-aggregation algorithms.

4.1 Experimental Model

Network and Traffic Model: The network topologies are randomly generated by GT-ITM [1]. Each node is treated as a router and has one host attached. The number of the groups is 50% of the number of the nodes and each group on average consists of 18 members, which are chosen randomly. We assume that the arrival of the packets forms a

[8] The interested readers can refer [4] for details of the algorithms.

Poisson process. Source of anycast packets are chosen randomly among $\frac{1}{3}$ of the hosts. We choose 2-path routing.

Performance Metrics: i) *Compression Ratio:* Recall that the compression ratio is defined as $CR = \frac{|A^p|}{m*n}$. It is used to measure the performance of the compression of routing tables. The baseline of the compression ratio is the lower bound $\frac{\Delta}{m*n}$, which is defined in Theorem 2. ii) *Average End-to-End Delay:* It is used to evaluate effectiveness of load balance. Generally speaking, the smaller the average end-end delay the network traffic suffers, the better load balance the network can achieve. We consider the average end-to-end delay in SSPF (shortest shortest path first) routing [7] as the baseline of the three group aggregation algorithms.

Evaluation Methods: Our algorithms work on the original routing tables including all shortest paths to all members of all anycast groups, which is produced by MIN_D algorithm [7].

4.2 Numerical Results and Observations

4.2.1 Compression Ratio

In this experiment, for networks with sizes 100, 200, 300 and 400 nodes, 20 random network topologies are generated respectively. We consider the top 10% routers in terms of their routing table sizes [9] in each network topology. The average compression ratio is shown in Figure 3. From Figure 3, we make the following observations: i) The

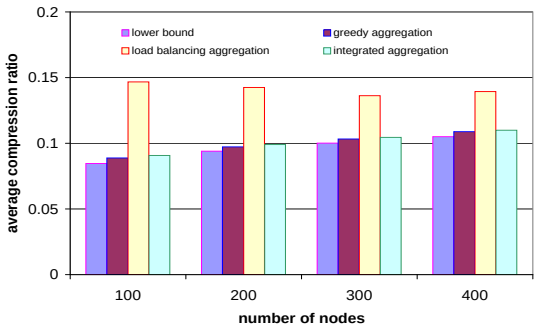

Figure 3. Compression Ratio.

greedy aggregation algorithm out-performs the other two algorithms and its compression ratio is close to the lower bound. This can be attributed to the fact that greedy aggregation algorithm always chooses the longest block in the synchronization matrix that induces minimun number of ranges. The load balancing algorithm is purely load balancing oriented, some non-zero entries, which are important to achieve high compression ratio, are reset by the load balancing algorithm. The extra-phase in integrated aggregation leaves the longer block. ii) The compression ratio is not very sensitive to the size of the network.

[9]The larger the routing table is, the more the group-aggregation is needed.

4.2.2 Average End-to-End Delay

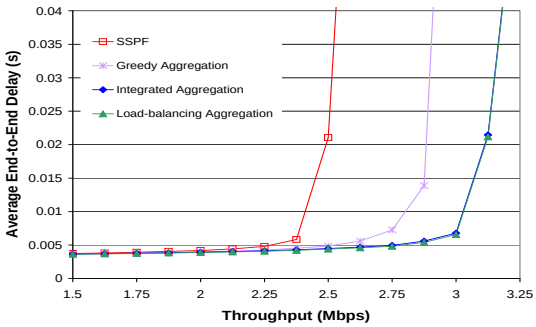

Figure 4. Average Delay.

In this experiment, 20 network topologies with 100 nodes are generated. The capacity of the links in these topologies is 10Mbps. The result showed in Figure 4 is from one of the network topologies. For other network topologies, we obtained the similar results. From Figure 4, we make the following observations: i) Multi-path routing with any one of the three aggregation algorithms out-performs the single path routing SSPF algorithm in terms of average end-to-end delay. ii) The average end-to-end delay in the system using the load balance algorithm is the lowest. The system using the integrated aggregation algorithm has a result, very close to the one obtained by the system using the load balance algorithm.

5 Final Remarks

In this paper, we propose a methodology of group-aggregation for scalable anycast routing. To the best of our knowledge, this is the first work in achieving scalability and high network performance in anycast routing by the aggregation approach.

The main idea of group-aggregation is to consider the treatment of inactive groups and alternative path selection as mentioned in subsection 3.1 while turning entries with same synchronization ID in any specific interface with the successive group IDs into one entry. Three group-aggregation algorithms have been proposed to achieve different goals: i) minimizing the routing table size; ii) balancing interface load; and iii) trade-off between minimizing routing tables size and balancing interface load. These algorithms take full advantage of the special semantics of anycast in interface determination and sychronization ID assignment. Sychronization ID is carefully assigned for exclusive forwarding in distributed router architecture to achieve high compression ratio on the routing tables. We have evaluated the

proposed algorithms in terms of the compression ratio and average end-to-end delay. The result shows that our group-aggregation algoritms can reduce routing table size by 90% while achieving very good network performance in terms of the end-to-end delay.

The work presented in this paper is the first step towards group aggregation for anycast routing. Some of the future researches are listed as follows: i) *Dynamic group aggregation*, where dynamics of anycast groups such as arrivals and departures of members in some anycast groups are considered, and ii) *Weighted interface determination*, where different interfaces may be assigned different weights for packet forwarding to achieve better performance, etc..

References

[1] http://www.cc.gatech.edu/projects/gtitm/.

[2] V. Fuller, T. Li, J. Yu, and K. Varadhan. Classless inter-domain routing (cidr): an address assignment and aggregation strategy. In *RFC 1819*, 1993.

[3] Ming-Huang Guo and Ruay-Shiung Chang. Multicast atm switches: Survey and performance evaluation. 28(2):98–131, April 1998.

[4] Zhibin Mai, Shengquan Wang, Dong Xuan, and Wei Zhao. Group aggregation for scalable anycast routing. In *Tech. Rep., Department of Computer Science, Texas A&M University*, 2002.

[5] S.Deering and R.Hinden. Internet protocol version6(ipv6) specification. In *RFC 2460*, 2000.

[6] David Thaler and Mark Handley. On the aggregatability of multicast forwarding state. In *Proceedings of INFOCOM*, pages 1654–1663, 2000.

[7] D. Xuan, W. Jia, W. Zhao, and H. Zhu. A routing protocol for anycast messages. In *IEEE Transactions on Parallel and Distributed Systems*, volume 11, pages 571 – 588, 2000.

Appendix A: Proof of *Theorem 1*

Given p-path sychronization matrix A^p, if the number of non-zero entries for each group is not less than p, *i.e.*, $\sum_{j=1}^{n} a_{ij} \geq p$ holds for $i = 1, 2, \ldots, m$, we can abtain a p-path range collection $RC = \{RC^k : k = 1, 2, \ldots, p\}$, where RC^k can be expressed as $RC^k = \{R^k[i_h^k, i_{h+1}^k; j_h^k] : h = 1, 2, \ldots, l^k\}$ for $k = 1, 2, \ldots, p$. Define i_h^k's (for $h = 1, 2, \ldots, l_k$, $k = 1, 2, \ldots, p$) as break points. In increasing order, sort them to form a partion

$$V = (v_1, v_2, \ldots, v_s), \tag{5}$$

where $s = \sum_{k=1}^{p} l_k$, and

$$v_1 = v_2 = \cdots = v_p = 1, \tag{6}$$

$$v_{s-p} = v_{s-p+1} = \cdots = v_s = n + 1. \tag{7}$$

Note that $|A^p| = s - p$, and if we want to minimize the size of the p-path sychronization matrix, it is equivalent to minimizing s. In the following, we want to show that our *greedy aggregation* algorithm achieves the smallest s. Given group-interface matrix A, assume that $V = (v_1, v_2, \ldots, v_s)$ is the partition for *greedy aggregation* algorithm and $\bar{V} = (\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_{\bar{s}})$ is the partition for arbitrary sychronization algorithm. We have the following lemma:

Lemma 1 *For the p-path range collection, as $i \leq \min\{s, \bar{s}\}$, we always have $v_i \geq \bar{v}_i$.*

Proof: We will use induction in terms of i. As we know that $v_1 = v_2 = \cdots = v_p = \bar{v}_1 = \bar{v}_2 = \cdots = \bar{v}_p = 1$, thus the inequality relation holds for $i = 1, 2, \ldots, p$. We want to show that if $v_r \geq \bar{v}_r$ for $r = i, i+1, \ldots, i+p-1$, then $v_{i+p} \geq \bar{v}_{i+p}$ also holds. Suppose that $v_{i+p} < \bar{v}_{i+p}$, we want to get contradictions.

Since there are $p + 1$ break points for p range subcollections, there must exist a range subcollection that includes [10] two of the break points $v_i, v_{i+1}, \ldots, v_{i+p}$. We choose such two break points v_{i+r_0} and v_{i+r_1} to construct range $R^{k^*}[v_{i+r_0}, v_{i+r_1}; j^*]$ such that $0 \leq r_0 < r_1 \leq p$ and $v_{i+r_0+1}, v_{i+r_0+2}, \ldots, v_{i+r_1}, \ldots, v_{i+p}$ are included in $p - r_0$ different range subcollections. Considering the p-path range collection obtained by arbitrary aggregation algorithm, we have the following cases as shown in Figure 5:

- $\bar{v}_{i+r_0}, \bar{v}_{i+r_0+1}, \ldots, \bar{v}_{i+p}$ *are included in different* $p - r_0 + 1$ *range subcollections:* By the arbitrary p-path range collection, we will find that there exist at least $r_0 + 1$ blocks $B[\bar{v}_{i+r_0}, \bar{v}_{i+p}; j_k], k = 1, 2, \ldots, r_0 + 1$ in A. Two of them can be got from $p - r_0 + 1$ range subcollections, and the other $r_0 - 1$ can be got from the other $r_0 - 1$ range subcollections.

- $\bar{v}_{i+r_0}, \bar{v}_{i+r_0+1}, \ldots, \bar{v}_{i+p}$ *are not included in different* $p - r_0 + 1$ *range subcollections:* It means that $\bar{v}_{i+r_0}, \bar{v}_{i+r_0+1}, \ldots, \bar{v}_{i+p}$ are included in at most $p - r_0$ different path selections. By the arbitrary p-path range collection, we will find that there exist at least $r_0 + 1$ blocks $B[\bar{v}_{i+r_0}, \bar{v}_{i+p}; j_k], k = 1, 2, \ldots, r_0 + 1$ in A. One of them can be got from $p - r_0$ range subcollections, and the other r_0 can be got from the other r_0 range subcollections.

In any case, we got $r_0 + 1$ blocks and these $r_0 + 1$ blocks may not be all included in different $r_0 + 1$

[10] Here we mean whether the corresponding end point of the given break point is in the specifi c range or not.

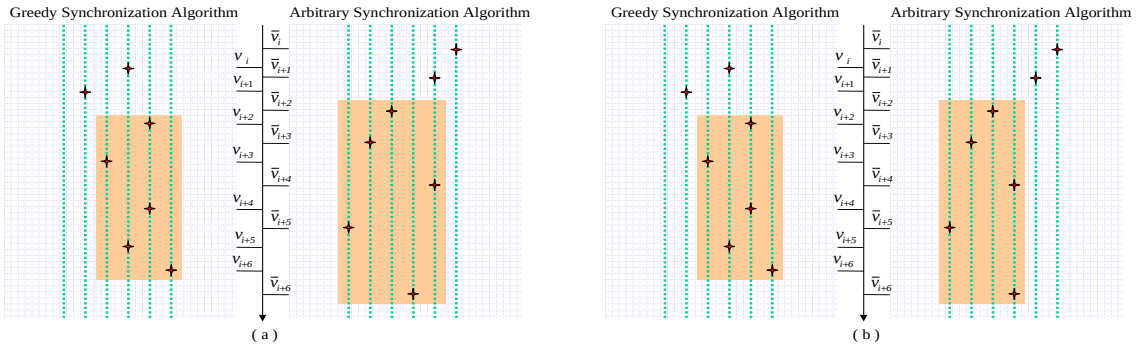

Figure 5. (a) and (b) are the cases that $\bar{v}_{i+r_0}, \bar{v}_{i+r_0+1}, \ldots, \bar{v}_{i+p}$ are included in different $p - r_0 + 1$ range subcollections or not respectively. Here is an example, where $r_0 = 0, r_1 = 2$, and $p = 6$.

range subcollections in greedy aggregation. It means that at least one is available. By greedy aggregation algorithm, range $R^{k^*}[v_{i+r_0}, v_{i+r_1}; j^*]$ will use subblock $B[v_{i+r_0}, v_{i+p}; j^{**}]$ of this block, but the range stops at v_{i+r_1} and the length of $R^{k^*}[v_{i+r_0}, v_{i+r_1}; j^*]$ is less than the one of $B[v_{i+r_0}, v_{i+p}; j^{**}]$. That is a contradicton.

In both cases, we have got contradictions. Thus, $v_{i+p} \geq \bar{v}_{i+p}$. By induction, $v_i \geq \bar{v}_i$ holds for $i \leq \min\{s, \bar{s}\}$. ■

Lemma 2 *For p-path range collection, $s \leq \bar{s}$ always holds.*

Proof: Suppose that $s > \bar{s}$, i.e., $s - p \geq \bar{s} - p + 1$, we have

$$v_{s-p} \geq v_{\bar{s}-p+1}. \tag{8}$$

By *Lemma 1*, we have

$$v_{\bar{s}-p+1} \geq \bar{v}_{\bar{s}-p+1}. \tag{9}$$

According to greedy aggregation algorithm, we have

$$v_s > v_{s-p}. \tag{10}$$

By (8), (9) and (10), we have

$$v_s > \bar{v}_{\bar{s}-p+1}. \tag{11}$$

However,

$$v_{s-p+1} = v_s = \bar{v}_{\bar{s}-p+1} = \bar{v}_{\bar{s}} = m + 1. \tag{12}$$

(12) contradicts to (11), therefore $s \leq \bar{s}$ must hold. ■

Let's begin to prove *Theorem 1*.

Proof: By *Lemma 2*, we have $s - p \leq \bar{s} - p$. Recall that $s - p$ and $\bar{s} - p$ are the size of the p-path synchronization tables for greedy aggregation algorithm and the arbitrary one respectively. Thus, the greedy aggregation algorithm achieves the minimum size of the p-path synchronization tables. ■

Appendix B: Proof of *Theorem 2*

Given group-interface matrix A, for each group i, $i = 1, 2, \ldots, m$, if $\sum_{i=1}^{n} \leq p$, append $p - n_i$ columns vector $e_i = (0, \cdots, 0, 1, 0, \cdots, 0)^{\perp}{}_{m \times 1}$ to A to form a augmented matrix $\hat{A}$. Note that each group has more than p non-zero entries. Then we can run greedy aggregation algorithm on $\hat{A}$ to get a p-path sychronization matrix. Remove the augmented interfaces from this sychronization matrix, then we can obtain a matrix $\hat{A}^p$, which is defined as a pseudo sychronization matrix since some small synchronization ID may be skipped in some group.

In the following lemma, given group-interface matrix A, we want to show that the size of any p-path synchronization matrix can be bounded by $|\hat{A}^p|$:

Lemma 3 *Assume that A^p is the p-path sychronization matrix obtained by arbitrary greedy aggregation, then*

$$|\hat{A}^p| \leq |A^p|. \tag{13}$$

Proof: Consider augmented matrix $\hat{A}$ of A, we can run the greedy aggregation algorithm on $\hat{A}$ to get a p-path sychronization matrix $\hat{A}^p_+$. On one hand, $\hat{A}^p_+$ is composed of two parts: $\hat{A}^p$ and $\mathbf{1}^p$ (all newly inserted entries with assigned synchronization IDs), then

$$|\hat{A}^p_+| = |\hat{A}^p| + |\mathbf{1}^p|. \tag{14}$$

On the other hand, with some modification [11] of sychronization IDs in $\mathbf{1}^p$, A^p and $\mathbf{1}^p$ can form a p-path sychronization matrix $\hat{A}^p_*$, then

$$|\hat{A}^p_*| = |A^p| + |\mathbf{1}^p|. \tag{15}$$

By *Theorem 1*, $\hat{A}^p_+$ is the p-path sychronization matrix with *minimum* size for $\hat{A}$, hence $|\hat{A}^p_+| \leq |\hat{A}^p_*|$. By (14) and (15), we have $|\hat{A}^p| \leq |A^p|$. ■

Set $\Delta = |\hat{A}^p|$, apply it into *Lemma 3*, then *Theorem 2* will be proved.

[11] The modification will not increase the number of ranges