AMBTS: A Scheme of Aggregated Multicast Based on Tree Splitting

Zhi-feng Liu, Wen-hua Dou, and Ya-jie Liu

School of Computer, National University of Defense Technology,
ChangSha, China, 410073
{liuzhif,liuyaj}@hotmail.com, wenhd@public.cs.hn.cn

Abstract. As the number of simultaneously active groups increases, IP multicast suffers from scalability problem. In backbone networks, the state scalability problem is even more serious. In this paper, we propose a novel scheme, called *AMBTS*, which uses the concept of aggregated multicast, and employs tree splitting before aggregating. We design an algorithm to divide the leaf nodes of backbone into different sub-groups, and then splitting the native multicast spanning tree. We propose a scheme to assign a set of aggregated multicast trees to deliver packets for a group. Analyses and simulations show that *AMBTS* can greatly improve state scalability: the number of aggregated trees is bounded in a small fixed number, and the multicast routing entries in transit nodes decreased dramatically.

1 Introduction

IP Multicast was first introduced in S. Deering's Ph.D. dissertation in 1988[1] and since then it has been an active area of research. However, the deployment of IP Multicast is limited and sparse due to a variety of technical and non-technical reasons. Among the issues which delay the deployment of IP multicast, state scalability is one of the most critical ones[2][3]. The state scalability concerns two aspects: scalability with regard to the number of receivers and scalability with regard to the number of simultaneously active multicast groups. In this paper, we only concern about the latter problem.

In traditional IP multicast, every on-tree node must maintain a forwarding state for each group (or even per-group/source), which increases linearly with the number of groups. On the Internet, the size of the multicast groups is limited only by the available address space: $2^{28}$for IPv4 and 2^{120} for IPv6. Growing number of forwarding states means not only more memory requirement but also slower forwarding process since the forwarding action of each packet involves an address looking up. Thus, multicast will suffer from scalability problems when the number of simultaneously active multicast groups is very large.

In unicast, the hierarchical address allocation structure [4] can lead to significant reduction in the unicast forwarding table. Because the address prefixes reflect the physical proximity of the network nodes. However, in multicast one cannot make any assumption about the location of receivers of a group, it is

N. Mitrou et al. (Eds.): NETWORKING 2004, LNCS 3042, pp. 829–840, 2004.

impossible to aggregate multicast forwarding table entries of different multicast groups.

In recent years, researchers have proposed some techniques to solve the forwarding state scalability problem. We classify the different proposals into four categories. In the first category, the schemes aim at eliminating the forwarding states from the routers. Application layer multicast [5][6][7] belongs to this class. In these architectures, the complexity is pushed to the end points. A lot of obstacles that confronted by IP multicast can be conquered. But there are some other problems arise in those protocols: they are less efficient than IP multicast. In the second category, the forwarding states at non-branched routers are reduced [8][9]. These techniques mainly target networks with a large number of sparse groups. In the third category, researchers try to aggregate forwarding states in routers using the same techniques as unicast [10][11].These techniques depend heavily on multicast address allocation. In the last category, multiple different groups are forced to use the same deliver tree, so that the number of deliver trees and forwarding states in backbone can be reduced [12][13]. A.Fei and J.H.Cui et al. proposed an algorithm to assign aggregated trees to multicast groups[12] and extensive simulations proved that aggregated multicast was a promising direction for scalable transit domain multicast provisioning. However, if the size of groups is as different as possible and the receivers of groups spread as diffuse as possible, the basic aggregated multicast scheme can do little for states reduction. This phenomenon will be seen in the following sections.

The rest of this paper is organized as follows: Section 2 presents an algorithm based on tree splitting for group-tree matching. Section 3 analyses the performance of *AM* (Aggregated Multicast) and *AMBTS* (Aggregated Multicast Based on Tree Splitting). Section 4 provides extensive simulations to study the performance of *AM* and *AMBTS*. Section 5 concludes our work.

2 A Scheme of Aggregated Multicast Based on Tree Splitting

2.1 Overview of AMBTS

To improve the state scalability of multicast in backbone domains, we propose a novel scheme of aggregated multicast based on tree splitting (*AMBTS*). The main idea is that we can classify the leaf nodes of the backbone into different subgroups in advance. As a multicast group comes into the backbone, we can divide the leaf nodes of this group into different sub-trees following the dividing scheme defined beforehand. We assign an aggregated tree for each sub-tree, and then we can get a set of aggregated trees for the native multicast group tree.

Every multicast group has a *native* multicast tree, which we denote as $T_0(G)$, the native multicast tree can cover the entire group receivers and never forward packets to those nodes without receivers. The native tree $T_0(G)$ can be calculated through some routing algorithm, such as *PIM-SM* [14]and CBT[15]. A splitting

of tree $T_0(G)$ can be defined as:

$$T_0(G) = \sum_{i=1}^{c} T_0^i(G) \ . \tag{1}$$

Where, c is the number of portions that the leaf nodes of backbone have been divided. If a set of aggregated trees $T_j, j = 1, 2, \ldots, c$, are used to replace the group's native sub-trees $T_0^j(G), j = 1, 2, \ldots, c$, the cost of bandwidth can be defined as:

$$\delta(G, \sum_{j=1}^{c} T_j) = \frac{C(\sum_{j=1}^{c} T_j) - C(T_0(G))}{C\,(T_0(G))} \ . \tag{2}$$

Where, $C(T_0(G))$ is the cost of tree $T_0(G)$. In intuition, $\delta(c_0, T)$ reflects the percentage of link overhead when we use aggregated sub-trees $T_0^j(G), j = 1, 2, \ldots, c$ to deliver packets for multicast group G.

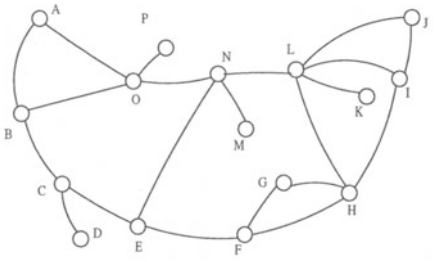

Fig. 1. vBNS backbone topology

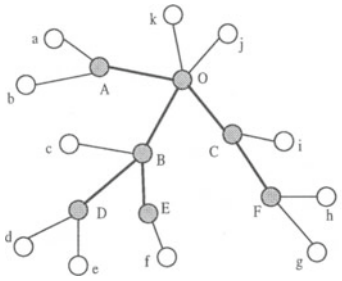

Fig. 2. A spanning tree rooted at node O

In order to compare the performance of AMBTS and AM, we do our analyses and simulations with a real network topology, vBNS IP backbone, illustrated in Fig. 1. In vBNS backbone, there are 16 core routers (they will not be terminal nodes for any multicast group), and we assume each of these core routers is attached with an edge router. Thus, there are totally 32 routers in the target network. The 16 leaf routers can join any multicast session. So, the number of different spanning trees rooted at any core routers in backbone is $2^{16} - 1$.

If we only allow perfect match, as the number of simultaneously active groups increases, the number of aggregated trees will be very large, even if we adopt tree aggregation. But, if we classify all the leaf nodes into different parts in advance, then the potentially number of trees that the backbone should maintain can be dramatically decreased. For example, if we classify the leaf nodes rooted at a core router into 2 sub-groups, 8 leaf nodes in each group, the number of different spanning trees rooted at this core router can not exceed $2(2^8 - 1)$.

In conclusion, if we classify the leaf nodes into several groups beforehand and then aggregate the sub-trees based on the splitting scheme, we can get only

a small number of aggregated trees. The number of aggregated trees can be bounded in a small fixed number when the simultaneously active groups become very enormous in backbone.

2.2 Classify Technique

In the following section, we will introduce a technique to split the leaf nodes. Using this technique, we will classify those leaf nodes into different sub-groups with similar size. It is not difficult to know that the number of different combinations of leaf nodes will be minimized if the size of each sub-group is equal.

Fig. 2. shows a spanning tree T built by PIM-SM or other core-based multicast routing protocol in a backbone network, the RP node is O. There are 6 core routers and 11 leaf routers. We use the term aggregated sub-tree leaf nodes (S_j) at router j to denote the entire set of leaf nodes served by all routers in the sub-tree rooted at j in T. The number of such aggregate sub-tree leaves, $s_j = |S_j|$ is given by:

$$s_j = \sum_{k \in children(j)} s_k$$
$$s_k = 1, \quad \text{if } k \in leaf(T) . \tag{3}$$

For example, in Fig. 2., $s_a = s_b = \ldots = s_j = s_k = 1$, $s_A = 2$, $s_B = 4$, $s_C = 3$, $s_D = 2$, $s_E = 1$, $s_F = 2$, $s_O = 11$.

In this example, node O has 5 direct children, i.e. there are 5 sub-trees rooted at O. We can set each of the 5 sub-trees into different sub-groups. If we classify the sub-trees into c sub-groups, the number of the different aggregated trees cannot exceed:

$$\sum_{i=1}^{c}(2^{n_i} - 1), \quad n_1 + \ldots + n_c = n \tag{4}$$

n_i is the number of leaf nodes in sub-group i.

The more parts we divide, the fewer trees we will maintain. On the other hand, the more parts that we divide, the more complexity that the RP node will involve. The reason is that each packet will encapsulate with different multicast address in RP node.

2.3 Algorithm Description

The deployment of aggregated multicast based on tree splitting includes two processes: foreclosing and group-tree matching. The foreclose process can be done offline. We describe the two processes in detail.

Foreclosing

1. Calculate a native multicast tree $T(r_i)$ rooted at some core router r_i. The spanning tree will cover all of the backbone leaf routers;

2. For each child node of r_i in tree $T(r_i)$, calculate the aggregated sub-tree leaves s_j, $j \in children(r_i)$;
3. Classify the leaf nodes into c sub-groups based on s_j, $j \in children(r_i)$, the number of leaf nodes in each sub-group should be as similar as possible;
4. For each potential root router r, repeat steps (1) to (3). After the foreclosing process, tree $T(r_i)$ is divided into c sub-trees: $T^1(r_i)$, $T^2(r_i)$, ..., $T^c(r_i)$.

Group-Tree Matching

1. According to foreclosing, we split the native tree $T_0(G)$ of group G into $T_0(G) = \sum_{i=1}^{c} T_0^i(G)$;
2. Find an appropriate match for $T_0^1(G)$, $T_0^2(G)$, ..., $T_0^c(G)$: for any sub-tree T_k that has the same root r in MTS(Multicast Tree Set), if $leaf(T_k) \supseteq leaf(T_0^j(G)), j = 1, 2, \ldots, c$, choose the one that can minimize $C(T_k) - C(T_0^j), j = 1, 2, \ldots, c$, denoted as $T_{j\_min}, j = 1, 2, \ldots, c$;
3. If there is not any sub-tree T_k rooted at r in MTS that can satisfy $leaf(T_k) \supseteq leaf(T_0^j(G)), j = 1, 2, \ldots, c$, $T_0^j(G)$ will be added to MTS;
4. Calculate the bandwidth overhead:

$$\delta(G, \sum_{j=1}^{c} T_{j\_min}) = \frac{C(\sum_{j=1}^{c} T_{j\_min}) - C(T_0(G))}{C(T_0(G))} \tag{5}$$

if $\delta(G, \sum_{j=1}^{c} T_{j\_min}) > bth$, bth is the allowed bandwidth overhead threshold, we will choose a sub-tree T_{i_min} from $T_{j\_min}, j = 1, 2, \ldots, c$ that can maximize $\delta(T_0^i(G), T_{i\_min})$. Then let the native sub-tree $T_0^i(G)$ be an aggregated tree, and add it to MTS, repeat step (4), until the set of sub-trees can satisfy $\delta(G, \sum_{j=1}^{c} T_{j\_min}) \leq bth$.

3 Performance Analysis of AMBTS

3.1 Performance Metrics

There are two natural metrics to evaluate aggregated multicast: the number of aggregated trees and the number of forwarding states in transit nodes.

Number of aggregated trees. (or "number of trees" for short) The number of aggregated trees is defined as $|MTS|$, where MTS is the current set of multicast trees maintained in backbone network. The more multicast trees, the more memory required and the more processing overhead involved.

Forwarding states in transit nodes. (or "transit states" for short). Forwarding states in terminal nodes can not be reduced in any multicast scheme, even in aggregated multicast. So we only measure the forwarding states in transit nodes.

3.2 Theoretical Analysis

To assist analyses. We list some parameters in table 1. In this section, we compare the performance between AM and $AMBTS$. We use NAT to represent the potential number of different aggregated multicast trees if perfect match allowed only. NAT is a stochastic variable, we can calculate the mean of it: $E(NAT)$.

Table 1. List of parameters

Parameters	Represent
l	the number of active groups
n	the number of leaf nodes
m	the number of RP routers
c	the number of divided portions of the backbone leaf nodes
p	session density
bth	bandwidth overhead threshold
L	the number of aggregated trees in MTS
N	the number of states in transit nodes

In AM, $E(NAT_{AM})$ can be calculated by formula (6):

$$E(NAT_{AM}) = m\sum_{i=1}^{c} C_n^i p^i (1-p)^{n-i} E(n,i)$$
$$E(n,i) = C_n^i \left[1 - \left(\frac{C_n^i - 1}{C_n^i}\right)^{n_i}\right]$$
$$n_i = \frac{l/(1-(1-p)^n)}{m} \times C_n^i p^i (1-p)^{n-i} \tag{6}$$

In $AMBTS$, $E(NAT_{AMBTS})$ can be calculated by formula (7):

$$E(NAT_{AMBTS}) = \sum_{k=1}^{m}\sum_{j=1}^{c_k}\sum_{i=1}^{n_{k,j}} C_{n_{k,j}}^i p^i (1-p)^{n_{k,j}-i} E(n_{k,j},i)$$
$$E(n_{k,j},i) = C_{n_{k,j}}^i \left[1 - \left(\frac{C_{n_{k,j}}^i - 1}{C_{n_{k,j}}^i}\right)^{n_{k,j}}\right]$$
$$n_{k,i} = \frac{l/(1-(1-p)^n)}{m} \times C_{n_{k,j}}^i p^i (1-p)^{n_{k,j}-i}$$
$$\sum_{i=1}^{c_k} n_{k,i} = n, \quad k = 1,2,\ldots,m \tag{7}$$

From formula (6)., it is clear that $E(NAT_{AM})$ associate with the number of RP nodes, session density and the simultaneously active groups. We can simplify formula (6). as:

$$E(NAT_{AM}) = f(l,m,p) \tag{8}$$

From formula (7)., $E(NAT_{AMBTS})$ associate with the number of RP nodes, session density, the simultaneously active groups and the number of portions that the spanning tree have been divided into. We can simplify formula (7). as:

$$E(NAT_{AMBTS}) = g(l, m, p, c) \tag{9}$$

Table 2. m=1, c=1 the relationship of $E(NAT_{AM})$ with l and p

$p\backslash l$	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
0.1	71	119	161	197	230	363	560	720	855	973
0.3	100	198	295	390	484	936	1772	2537	3247	3911
0.5	100	200	300	400	499	994	1971	2934	3882	4815
0.7	100	198	294	389	482	933	1767	2530	3238	3901
0.9	61	102	138	171	200	318	493	635	757	864

Table 3. m=1, c=2 the relationship of $E(NAT_{AMBTS})$ with l and p

$p\backslash l$	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
0.1	54	76	92	104	112	144	184	206	222	236
0.3	132	202	250	284	312	388	448	472	484	492
0.5	168	280	356	406	440	500	510	510	510	510
0.7	132	202	250	284	310	388	448	472	484	492
0.9	48	70	84	96	104	134	172	196	212	224

We calculate $E(NAT)$ using formula (6). and (7) with different p, c, l and node B as RP node in Fig. 1. The result is shown in table 2. and 3. From these tables and formula (6).to(7)., we can see:

1. With the increase of the active groups, $E(NAT)$ increases;
2. $E(NAT)$ reaches maximum while p equals to 0.5. As p gets away from 0.5, $E(NAT)$ decreases;
3. $AMBTS$ superior to AM when l is a larger number;
4. For any m, p and c, we can find a critical number l_c, l_c is the critical performance point of AM and $AMBTS$:

$$\begin{aligned} f(l, m, p) &< g(l, m, p, c) \quad l < l_c \\ f(l, m, p) &= g(l, m, p, c) \quad l = l_c \\ f(l, m, p) &> g(l, m, p, c) \quad l > l_c \end{aligned} \tag{10}$$

l_c decrease as c increase if m and p unchanged.

4 Simulations and Analysis

Given the lack of large scale multicast experimental traces, we use the group model developed in [12]: the random node-weighted model. In this model, each node is assigned a weight representing the probability that the node belongs to a group. Then we can control the size of the group, in other words, we can control the session density [3]. We assume every node have the same probability to be in each group and change this probability to find out the relevance between session densities with the number of aggregated trees.

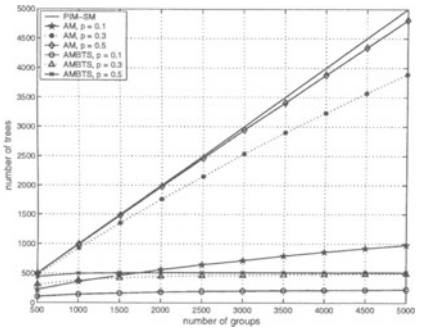

Fig. 3. L vs l, bth=0, c=2, m=1

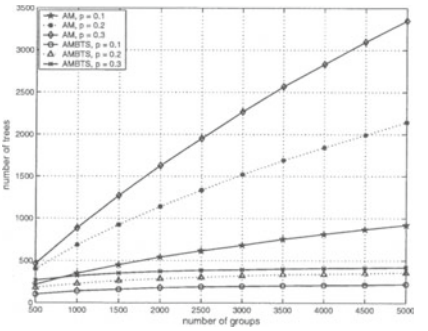

Fig. 4. L vs l, bth=0.1, c=2, m=1

We design two set of experiments to compare *PIM-SM* and *AM* and *AMBTS*. In the first group, there is only one core router selected as *RP* node in backbone. In the second group, three core routers are used as *RP* nodes in order to achieve better load balancing.

In the first set of experiments, the *RP* node is B, and the leaf routers in backbone are divided into 2 groups according to the foreclosing process presented in Sect.2., and the group-tree matching algorithm introduced in Sect.2. is used. Correspondingly, the routing algorithm is *PIM-SM* like routing algorithm, which uses unidirectional shared tree.

Fig. 3. shows the relevance of the number of aggregated trees with the simultaneously active groups when we only allow perfect-match. We can see that in *PIM-SM* the number of trees is a linear function of the number of active groups. If we use the basic *AM*, the number of aggregated trees is quite different as the session density changes. In contrast, if we use *AMBTS*, although the number of aggregated trees increases with the session density, the range is limited. As the session density is 0.1 and the concurrent groups increase from 500 to 5000, the number of aggregated trees only increases from 104 to 224. The influence of session density is less than that in AM. When session density is 0.5 and active groups are 5000, there are 510 aggregated trees in backbone, which is just as analyzed before.

Fig. 4. compare the performance of *AM* and *AMBTS* when we allow pure-leaky match. Fig. 4. plots the relationship between the active groups and the aggregated trees when the bandwidth overhead threshold is 0.1. In this Fig., we can see that both *AM* and *AMBTS* can get more aggregation. In *AM*, when session density is 0.1 and the active groups are 5000, we can get 5.8% aggregation more than that in perfect-match. If the session density increases to 0.3, there are 4129 aggregated trees, decreased about 14.2% than that in perfect-match. In *AMBTS*, when session density is 0.1 and the number of active groups is 5000, we need to maintain 221 aggregated trees. When the session density increases to 0.3, there are only 358 aggregated trees, decreased about 29.8% compared to the perfect-match. The similar result can be attained if we change the bandwidth overhead threshold. In a word, increasing the bandwidth overhead threshold can improve the ability of aggregation.

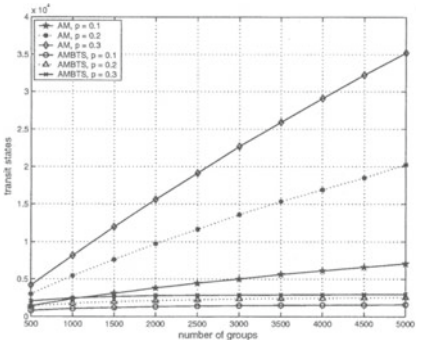

Fig. 5. N vs l, bth=0, c=2, m=1

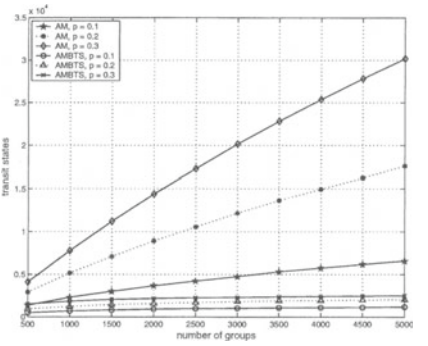

Fig. 6. N vs l, bth=0.1, c=2, m=1

Fig. 5. shows the relevance of the number of forwarding entries in core routers and the number of active groups when we only allow perfect-match. From this Fig. we can see that if we use the *AMBTS*, the number of forwarding entries in core routers is much less than that in *AM*. There are two reasons: first, in *AMBTS*, there are less aggregated trees; second, the leaf nodes in aggregated trees of *AMBTS* are less than that in aggregated trees of *AM*. For these two reasons, *AMBTS* is superior to *AM* in aspect of the number of forwarding entries in core routers. In Fig. 5., when session density is 0.3 and the number of active groups is 5000, there are 35202 forwarding entries in core routers if we use *AM*. In contrast, if we use *AMBTS*, the number of forwarding entries in core nodes is only 2988, it is only 8.5% of *AM*'s.

Fig. 6. compares the performance of *AM* with *AMBTS* in the number of forwarding entries in transit routers when we allow pure-leaky match. We obtain the similar trends with Fig. 5. The more we allow the bandwidth overhead, the less number of forwarding entries we should maintain in transit routers. The reason is that more and more groups can use the same aggregated tree.

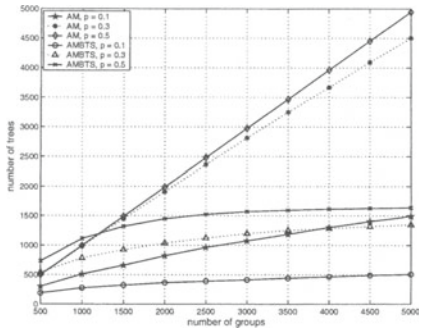

Fig. 7. L vs l, bth=0, c=2, m=3

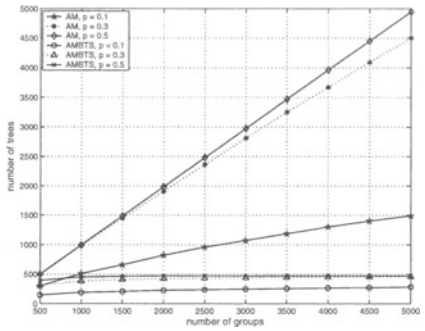

Fig. 8. L vs l, bth=0, c=3, m=3

Compare to *AM*, *AMBTS* can get even more aggregation. With the increase of simultaneously groups in backbone, the number of forwarding entries maintained in transit routers close to a smaller number, which is owing to the stability of the smaller number of aggregated trees.

In the second set of experiments, there are three core routers E, H and N chosen as RP routers. When a multicast session starts up, its RP node is randomly chosen from the 3 RP routers. The leaf routers in backbone are divided into 2 or 3 groups.

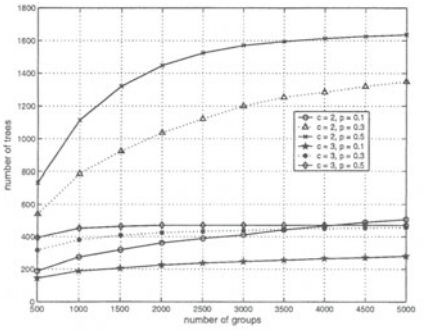

Fig. 9. L vs l, bth=0, c=2 or 3, m=3

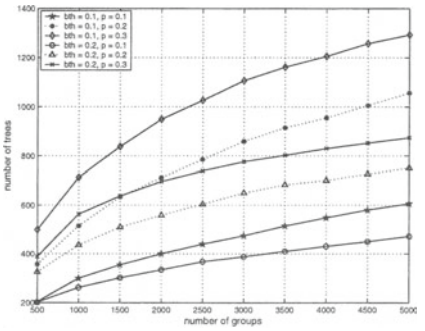

Fig. 10. L vs l, bth=0.1 or 0.2, c=2, m=3

Fig. 7. shows the relevance of aggregated multicast trees with the number of active groups when the number of RP nodes great than 1. In this experiment, the leaf routers are divided into 2 groups. In this Fig., we can find that the limitation of *AMBTS*, the aggregated multicast trees may be small if the number of active groups is large, but there are possibly more aggregated trees than *AM* when the number of active groups in backbone is not large enough. For example, *AM* superior to *AMBTS* when $l < 1200$, $p = 0.5$, $c = 2$ and $m = 3$. If we increase

c, the critical point l_c will decrease dramatically. In fig. 8., $p = 0.5$, $c = 3$ and $m = 3$, *AMBTS* is superior to *AM* as l exceeds 500.

Fig. 9. shows the relevance between the number of divided portions with the number of aggregated trees. From this figure. we can see that increase the number of divided portions will decrease the number of aggregated trees in different group session density. When $c = 2$, $m = 3$, and the number of active groups is 5000 there are 1636 aggregated trees. However, if $c = 3$, and the other parameters keep same, there are only 471 aggregated trees. Namely, in the second case, there are only 28.7% of trees need to maintain compare to the first case.

Fig. 10. and Fig. 11. compares the performance of *AMBTS* that in different bandwidth overhead threshold when the number of *RP* routers is 3. The trends are similar to the result of the first set of experiments. The difference is that there are more aggregated trees and more transit states in each circumstance.

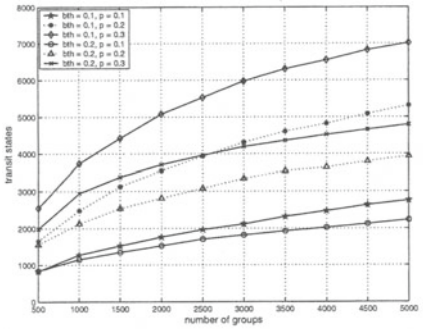

Fig. 11. N vs l, $bth = 0.1$ or $o.2$, c=2, m=3

5 Conclusions

In this paper, we propose a novel aggregated multicast scheme based on tree splitting, called *AMBTS*. This scheme can dramatically reduce the number of aggregated trees in backbone, and can take full advantage of aggregated multicast. In this paper, we design a tree splitting scheme, and a group-tree matching algorithm based on tree splitting. Through theoretical analysis and extensive simulations, we compare *AMBTS* with traditional multicast scheme and the basic *AM*. The simulation results show that *AMBTS* can obtain much more reduction in the number of aggregated trees and the number of forwarding state in core routers than the basic *AM* can get. The number of aggregated trees in backbone can be bounded in a small fixed number and this number can be calculated in advance, no matter how many simultaneous groups in the backbone. Thus, we can conclude that aggregated multicast based on tree splitting is a very promising scheme for transit domain multicast provision.

References

1. S. Deering: Multicast Routing in a Dategram Internetwork, Ph.D. thesis, Stanford University, 1991.
2. K. Almeroth: The evolution of multicast: From the MBone to inter-domain multicast to Internet2 deployment. IEEE Network, Jan./Feb. 2000.
3. T. Wong and R. Katz: An Analysis of Multicast Forwarding State Scalability, in Proceedings of the 8th IEEE International Conference on Network Protocols (ICNP 2000), Osaka, Japan, November 2000.
4. Y. Rekhter and C. Topolcic: Classless Inter-Domain Routing (CIDR). RFC1520, Septmember 1993.
5. D. Pendarakis, S. Shi, D. Verma, and M.Waldvogel.: ALMI: An application level multicast infrastructure. Proceddings of 3rd Usenix Symposium on Internet Technologies and Systems(USITS 2001), Mar. 2001.
6. M. Castro, P. Druschel, A-M. Kermarrec, and A. Rowstron.: SCRIBE: A large-scale and decentralized application-level multicast infrastructure. IEEE Journal on Selected Areas in communications(JSAC), 2002.
7. Y. Chu, S. Rao, and H. Zhang.: A case for end system multicast. Proceedings of ACM Sigmetrics, June 2000.
8. L. H. M. Costa, S. Fdida, and O. C. M. Duarte: Hop-by-hop multicastrouting protocol. Proceddings of SIGCOMM'01, Aug. 2001.
9. J. Tian and G. Neufeld: Forwarding state reduction for sparse mode multicast communications. Proceedings of IEEE INFOCOM, Mar. 1998.
10. P. I. Radoslavov, D. Estrin, and R. Govindan: Exploiting the bandwidth-memory tradeoff in multicast state aggregation. Technical report, USC Dept. of CS Technical Report 99-697 (Second Revision), July 1999.
11. D. Thaler and M. Handley: On the aggregatability of multicast forwarding state. Proceedings of IEEE INFOCOM, Mar. 2000.
12. A. Fei, J.-H. Cui, M. Gerla, and M. Faloutsos: Aggregated Multicast with inter-group tree sharing. Proceedings of NGC2001, Nov. 2001.
13. J.-H. Cui, J. Kim, D. Maggiorini, K. Boussetta, and M. Gerla: Aggregated Multicast — A Comparative Study. Accepted for publication in the special issue of Cluster Computing: The Journal of Networks, Software and Applications, Baltzer Science Publisher, 2003
14. D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei: Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification. IETF RFC 2362, June 1998
15. A. Ballardie: Core Based Trees (CBT version 2) multicast routing protocol specification. IETF RFC 2189, Sept. 1997