An Approach to Construct the Application Layer Multicast Overlay with Real-time Network Structure Detection

XING Ming-ming1, XING Ming-ming2, XING Ming3

(1.*Dept. Of \*\*\*\*, University, City Province Zip Code, China*；2. *Dept. Of \*\*\*\*, University, City Province Zip Code, China*；3. Dept. of \*\*\*\*, University, City Province Zip Code, China)

**Abstract:** This paper analyzed the path-driven model and data-driven application layer multicast system. In order to increase the network resource usage of each node, this paper constructs the application layer multicast system with network structure detection. This scheme detected the regional network structure of each node before constructing the multicast network. In this way, the new node can modify the multicast network to be more effective by self-adapting. At last, this paper will make several simulated tests to check the performance of this model.

**Key words:** application layer multicast; multicast network structure detection; self-adapting;

1. Introduction

Large-scale live video broadcast has been a very potential application on the Internet. However, there is no perfect solution to support this project so far. At first, researchers try to reach this target by using IP multicast. IP multicast has a great performance on efficiency, but at the same time it also strongly depends on the infrastructure and is difficult to widely deploy.

In consideration of the difficulty of the IP multicast deployment, researchers try to use the application layer multicast (ALM) instead of IP multicast. Application layer multicast does not need to change the network infrastructure. This method implements multicast forwarding function at the end-hosts. Although the application layer multicast is easy to deploy, the performance of application layer multicast is hard to control as the variability of network environment. How to construct the ideal multicast overlay better and fast becomes the current point of research.

1. Related Work

In the former research, the application layer multicast can be divided into three classes: path-driven model, data-driven model and combination of path-driven and data-driven model.

Path-driven model delivers data partly like IP layer multicast. In this model, the data travels through a tree-like network in which the source node acts as a root node and the other nodes act as non-root nodes. There are lots of protocols using path-driven model, such as NICE [1] designed by Banerjee at the University of Maryland and ZIGZAG [2] devised by Tran in University of Central Florida Orland. Otherwise, the SplitStream [15] which is devised by Castro is also a protocol using path-driven model. SplitStream is different from the first two, since SplitStream uses multi-tree path to deliver data with multiple description coding (MDC [16]) and the first two use a single-tree path. In these protocols, the single-tree model is easy to be managed. But this model does not use the upload bandwidth of leaf nodes, so it requires that each non-leaf node has a high upload speed. On the contrary, the multi-tree model has a high upload bandwidth utilization of leaf nodes, so it requires relatively low upload bandwidth of each node. But at the same time, this model is harder to maintain and is difficult to be optimized.

In consideration of the heavy dependency of path-driven model on a structured delivery path, this model is difficult to be optimized when the network situations of nodes are changeable. To solve this problem, researchers introduce data-driven model into ALM which is used to share files in peer-to-peer applications. The protocols of Coolstreaming [7-9] designed by Zhang Meng and the popular application such as PPLive and PPStream use this model to construct their multicast network overlays. The data-driven model does not depend on structured delivery path and has no need to maintain the complex delivery tree. But the delivery process in data-driven model contains two steps: request and response. To get a data block in data-driven model, node needs to send a request message and then get this data block from the requested node. So in this model the delivery efficiency is relatively low.

Since the advantages of data-driven model and path-driven model are complementary, researchers try to combine these two models to create a new delivery model. Many protocols try to improve their performances in this way, such as mTreebone [14] devised by Feng Wang and GridMedia [10-12] designed by Zhang Meng. MTreebone system improves its timeliness by using the path-driven model to construct its backbone network and using the data-driven model to support the leaf nodes. GridMedia try to improve its adaptability by converting steady data delivery links to fixed paths.

1. Analysis and Design

The previous researches try to construct an ALM overlay by combining data-driven model and path-driven model. Most of them try to use both model at same time and this makes the system complicated. Also, the disadvantages of both models cannot be avoided in this way. The solution presented in this paper tries to detect the multicast overlay in a way like data-driven model and then builds a path-driven delivery overlay. In this way, the ALM network may have a good performance of both efficiency and timeliness.

* 1. Node Degree and Delivery Delay

To construct an optimized AML overlay it is necessary to know the determinant factors of multicast overlay performance: the delivery delay and associated variables.

Delivery delay means the lag when a multicast data delivery from the server node to client node. For each single data packet, it will be delivered to every client node through a tree-like path whether in data-driven model or path-driven model. So the hop number of each packet satisfies the equation:

(1)

In this equation, argument *n* means the number of all nodes. Argument *k* is the average node degree. In the other word, *k* means how many child nodes each node can handle. Finally, the *hop* means the hop number of each packet.

Taking the transmission delay between nodes into consideration, the delivery delay can be calculated with the following equation:

(2)

In this equation, argument *delay* means the delivery delay a packet delivered from the server node to a client node and argument *d* means the average network transmission delay between a client node and its parent node. Considering the equation (1) and (2) simultaneously, the maximum delivery delay of one packet should satisfy the following equation:

(3)

In the equation (3), the argument *d* is hard to be modified and the argument *n* is determined by the number of client nodes, so this paper will provide an ALM constructing model to maximizing the argument *k* without raising the upload bandwidth requirement.

* 1. Model Design

In the above analysis, it can be concluded that when the node number is certain, to increase the degree of each node can improve the performance of multicast overlay network. Therefore this paper presents an adaptive overlay network constructing model called ASD model(adaptive ALM overlay with overlay structure detection).In this section, the ASD model will be described in three aspects: data splitting, adaptive joining process and structure detection.

* + 1. Data splitting

Usually, there are two methods to increase the degree of one node, increasing the upload bandwidth and reducing the stream rate. Since in the real network, the upload bandwidth is limited by the network environment, reducing the stream rate shall be the better method. To reduce the stream rate, we split the whole stream into several sub-streams. In this way, the data stream can be delivered as shown in fig. 1.



1. data delivery

In fig. 1, the node S is the server node and the number on the arrows means the hop number of the data. Without splitting the stream, the data was delivered to node 3 as shown as (a) in fig. 1. It takes 3 hops to reach node 3. By splitting the stream into two half rate sub-streams, one node can deliver certain sub-stream for two different child nodes, so the data can be delivered like (b) in fig. 1.It just took 2 hops to reach node 3.

In view of the continuity of the media stream, this paper splits the whole data stream by the serial number of each date packet, just as in the fig. 2.



1. data splitting

In fig. 2, each data packet of the original data was marked with a serial number. If the remainder, when the serial number is divided by 2, equals 1, the packet will be add to data sequence 1, else it will be add to data sequence 2.

* + 1. Adaptive Joining Process

Since the stream is split into several sub-stream, the managing of the multicast overlay become complex and inefficient. To solve this problem, the nodes in this paper will take an adaptive method to join the ALM network.

In past researches, if the stream was split into several sub-streams, these sub-streams will be treated as unrelated objects or they will be managed with a constant structure. The previous method makes the situation too complex to be implemented, because to manage a single tree is complex enough and this method needs to manage a multiple one. The later one is relatively simple, but it makes the ALM overlay hard to be modified or optimized.

To simplify the joining process and make the optimization easier, this paper makes each node manage the joining process by the node itself. To reach this point, a node needs several strategies to meet different situations. In following paragraphs, all these situations will be described.

1. Situation 1 is that the server node has some idle resources to handle new nodes. In this situation, the node will just become the child node of the server node.

2. Situation 2 is that a client node has some idle resources to handle new nodes. But different from the first situation, new node cannot simply become the child node of this client node, since this situation can be further divided into two different situations as situation 2a and 2b.

2a. Situation 2a means that the client node which has free resources is just delivering one kind of sub-streams as node 1 or node 2 that shown in (b) of fig. 1. In this situation, making the new node become a child node of this client node can maximize the delivery degree of this kind sub-stream.

2b. Situation 2b means that the client node which has idle resource is delivering two different kinds of sub-streams as node B shown in (a) of fig. 3. To maximize the output degree of each sub-stream, the new node should replace this client to handle certain sub-stream just as node D shown in (b) of fig. 3. In this way, the output degree of each sub-stream can be improved from 1 to 2. And the node D and node B can just handle one kind of sub-stream, so the output degree of each sub-stream can be optimized.



1. situation 2b

3. Situation 3 is that none of the joined node has free resources to handle this new node just as (a) of fig. 4. In past, this situation may mean there isn't any other new node that can join this multicast network. To deal with this situation, node will use the insert operation to join the multicast network as shown as node 3 in (b) of fig. 4. In fig. 4, both of node 1 and node 2 has a poor upload bandwidth, so they cannot handle a single sub-stream for more than one child node. But the new node 3 has a normal upload bandwidth and can handle two or more sub-streams. So node 3 is inserted between node 1 and its parent node to get one sub-stream, then it should be inserted between node 2 and its parent at the same time to get another sub-stream. In this way, node 3 can join the multicast network successfully. If unfortunately the new node also has a poor upload bandwidth, there is no way to make it join the multicast network, because the total upload bandwidth of all nodes should not smaller than the bandwidth they need.



1. situation 3
   * 1. Structure Detection

In section 3.2.2, this paper presents an adaptive method to manage the joining process of each node and this method depends on the regional overlay structure of the joined nodes. So in this section, this paper will describe the method of network structure detection.

To detect the network structure, the most important thing is to confirm what kinds of information are necessary. From the analysis in the above section, the information of each node used in the joining process includes the idle output degree, the parent node and child nodes of each sub-stream. Adding some necessary information to identify the node, the network structure detection packet can be drawn as the one in fig. 5.



1. structure detection information

This paper uses data-driven mode to get these packets, just as shown in fig. 6.



1. structure detection process

In fig. 6, node 4 will get the structure information in four steps:

1. The node 4 should ask the server node for the member list of current multicast network.

2. After send the asking message, node 4 will receive a response message from the server node.

3. Then node 4 can send a structure detection request to each member.

4. Finally, node 4 could receive several response messages from the members and can start joining process by using the information from these messages.

* + 1. Algorithm description

The description of the structure detection algorithm is shown in fig. 7.

The description of the adaptive joining algorithm is shown in fig. 8.

Structure detection algorithm:

1. //Initialize the nodes
2. NodeInit()
3. //Query the member list from server node.
4. SendQueryMsgToServerNode()
5. GetMemberInfoFromServerNode()
6. For each(member)
7. Do{
8. //Get the structure information from other members
9. SendEvalMsgToEachMember()
10. GetEvalResponseMsgFromEachMember
11. //Start the joining process if get enough information
12. IF(isEnoughInfo)
13. break
14. }
15. StartJoin()
16. description of the structure detection algorithm

Adaptive joining algorithm:

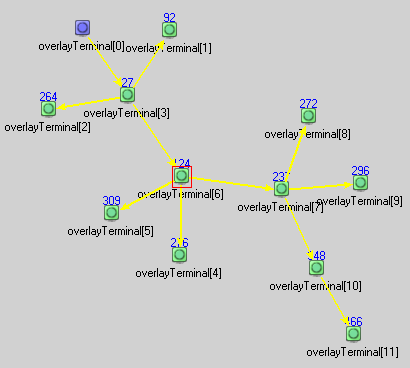
1. //The joining algorithm will be run for each sub-stream
2. ForEach(DataNo.)
3. Do{
4. //If the node has got this sub-stream already, the joining algorithm will be run for next sub-stream
5. If(ParentState[DataNo.]!=Init)Continue;
6. //firstly, the free resource of the server node will be used.
7. If(ServerNodeHasFreeResource){
8. UseServerAsParentNode();
9. }
10. //secondly, new node will chose the joined node which delivers the selected sub-stream specially.
11. ForEach(PeerMember)Do{
12. If(ChosenPeerHasFreeResource()&&
13. ChosenDataHasTheHighestUsage())
14. SelectTheChosenPeerAsParentNode();
15. }
16. //thirdly, new node will replace a joined node which delivers two different sub-streams at same time to handle certain sub-stream, as shown as situation 2b.
17. ForEach(PeerMember)Do{
18. If(IsPeerDeliverDifferentData){
19. UseTheParentNodeOfThisPeerAsParent();
20. ReplaceThisPeerToDeliverThisData()
21. }}
22. //if this new node has not get the selected sub-stream yet, it will use any free resource if there
23. ForEach(PeerMember)Do{
24. If(PeerHasFreeResource){
25. UserThisPeerAsParent();
26. }}
27. //if there no free resource anywhere, the new node will use the insert operation to make it join successfully if it can do, as shown as situation 3
28. ForEace(PeerMember)Do{
29. If(PeerDoNotDeliverThisData){
30. UseTheParentNodeOfThisPeerAsParent();
31. DeliverThisDataToThisPeer();
32. }}}
33. description of the adaptive joining algorithm
    1. Model Improvement

To verify the feasibility of ASD model, this paper made a small test. The result shows that the ASD model can make the output degree of each sub-stream reach the maximum, but it is bad at branch balancing.

* + 1. The problem of ASD model

The reason that ASD model has bad performance at branch balancing is that the new node always tries to choose the joined node which delivers the selected sub-stream specially. On the other way, this means if a node gets a sub-stream and does not deliver it, this node will never deliver this sub-stream to other node even no matter how early it get this sub-stream.

Since the reason above, the deliver path of certain sub-stream in the small test is shown as fig.9.



1. the deliver path of certain sub-stream in small test of ASD model

In fig. 9, the dark color node marked with overlay Terminal [0] is serve node and the light color nodes marked with overlay Terminal [1~11] are peer nodes. The number over the top of node is the timestamp which stands for the delay of the sub-stream. In this Figure, the overlay Terminal [1] and overlay Terminal [2] get this sub-stream early, but they do not deliver this sub-stream any more. The last node overlay Terminal [11] do not choose neither of them but chose overlay Terminal [10] as its parent node, even if overlay Terminal [10] needs 4 hops and 348ms to get this sub-stream.

This paper uses the low-timestamp first insert operation to avoid the situation shown in fig. 9 and presents a grade algorithm to decide when the insert operation shall be used. This new model will be called ASD-TS model in this paper.

* + 1. Low-timestamp first insert operation

In ASD model, the reason unbalances the deliver path is that the new nodes always avoid to choose the node that does not deliver the sub-stream as its parent node. So ASD-TS model will use low-timestamp first insert operation to make new nodes try to choose the node which gets the sub-stream early but does not deliver it anymore.

The process of the low-timestamp first insert operation is shown as fig. 10.



1. Low-timestamp first insert operation

In fig. 10, node S means server node and the nodes marked from 0 to 8 means the joined nodes. In this Figure, node 9 is a new node that tries to join this ALM network. Further, the upper node means it could get the sub-stream early, in other words, this node has a lower timestamp. These nodes in fig. 10 can be divided into two types. These nodes circled up by solid line deliver the sub-stream which is delivered through the path shown in fig. 10 and the other nodes circled up by dotted line deliver the other kind sub-stream which is not shown in fig. 10.

If in ASD model, when node 9 tries to join this ALM network to get the sub-stream shown in fig. 10, it may node 3 as its parent node, because node 3 has free resource and delivers the sub-stream which node 9 needs to get. But this is not a good choice, since the node 2 has a lower timestamp, and if the new node always does not choose node 2 as its parent node, the lower timestamp of node 2 will be wasted.

But if node 9 uses the low-timestamp first insert operation to join this ALM network, it will choose to insert between node 2 and the server node as shown as fig. 10(a). In this way, node 9 could get the sub-stream earlier and deliver this sub-stream to other nodes. And the final result is as shown as fig. 10(b).

* + 1. Grade algorithm

Since the low-timestamp first insert operation depends on the timestamp of each node, so it is important to decide that whether use this operation. So this paper presents a grade algorithm to tell whether to use this operation or not.

To devise the grade algorithm, the first thing is to know what operations are necessary and the effects of each operation.

From the description in the previous section, the operation can be divided into 4 types: join, switch, insert and low-timestamp first insert.

1. When a new node uses join operation to get a sub-stream, like situation 1 or situation 2a, the only effect is that this node will cost some resource of the selected parent node.

2. When a new node uses switch operation to get two sub-streams, like node D does in situation 2b, the effect is that the node D get these two sub-streams and he output degree of each sub-stream can be improved from 1 to 2.

3. When a new node uses insert operation to get a sub-stream, like node 3 does in situation 3, the effect is increase the timestamp of node 1 and node 2. And if node 1 or node 2 has child node, the timestamps of these child node will also be increased.

4. The effect of low-timestamp first insert operation is like insert operation, but this operation can make the deliver path more balanced.

So the switch operation is the good choice if it can be used and the join operation is the second choice if some node has free resources. And if some node get a sub-stream early and does not deliver it anymore the low-timestamp first inset operation should be used. And taking account of the timestamp, the grade algorithm of each operation is shown in fig. 11.

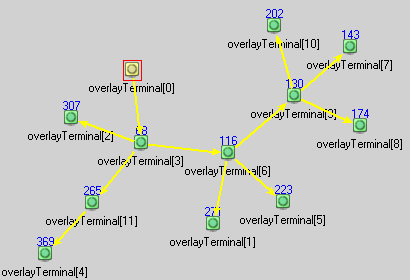
grade algorithm:

1. //The grade algorithm will be run for every peer in the network
2. ForEach(Member){
3. //The join operation only can be used when the peer has free resource
4. if(MemberHasFreeResource)
5. //the score of join operation depends on the timestamp and the which sub-stream this peer deliver
6. joinScore=dataRadio[dataNo.]/Timestamp[dataNo.] + lag);
7. else
8. joinScore=0;
9. if(MemberDeliverTwoKindsData)
10. //the score of switch is the average of both sub-stream
11. switchScore= (1/(Timestamp[dataNo.1]+lag) + 1/(Timestamp[dataNo.2]+lag)/2;
12. else
13. switchScore =0;
14. //the insert operation only can be used when the peer does not deliver the selected sub-stream
15. if(dataRadio[dataNo.]==0)
16. //k means the outdegree of this peer.
17. insertScore=1/(k\*(Timestamp[dataNo.] + lag));
18. else
19. insertScore=0;
20. }
21. description of grade algorithm
    * 1. Join process of ASD-TS model

By using the grade algorithm, the new node can choose a best peer and a best operation to get the sub-stream which it needs. So the join process of ASD-TS model should be divided into 3 steps.

The first step is calculating the score of all operations for each peer. Then the maximum score should be found out. Finally, the new node needs to use the selected peer and operation to join the ALM network.

The result of the small test is shown in fig. 12.



1. the deliver path of certain sub-stream in small test of ASD-TS model

From the result in fig. 12, it could be obviously seen that overlay Terminal [11] chose overlay Terminal [3] as its parent node and the maximum timestamp reduce from 466ms to 369ms.

1. Simulation and Analyses

To judge how a model works, the two main factors are feasibility and performance. Feasibility means the model is correct and the performance is good. This paper will run several simulations to do this work.

* 1. Simulation Environment

In this paper, OMNet++ [18] will be used for analysis feasibility and performance of ASD model and ASD-TS model. The codes of these simulations are based on the OverSim [17] project. The version of OMNet++ used here is 4.1 and the OverSim code package used here is OverSim-20101103 which was published on November 3, 2011.

* 1. Feasibility Analysis

The feasibility analysis in this paper includes two parts: small-scale simulations to view result intuitively and a large-scale simulation to prove the expansibility. In small-scale simulations, 12 nodes will be simulated and the upload bandwidth of each node is just equal to the download requirement. The data stream in this simulation will be split into three individual streams. This means that each node can handle three child nodes at most.

The results of the small-scale simulations are already shown before this section as fig.9 and fig.12. From the results we can see that every node has connected to the network, and the delivery paths are correct. These results prove that ASD model and ASD model are able to construct the ALM overlay.

To make further analysis, the relatively large-scale simulation should be taken. In the large-scale simulations, every simulation parameters are the same as the small-scale simulation except that the number of nodes becomes 500.The result shows that all 500 nodes start joining process at 493.2s, and this process was finished 150.2s later at 643.4s. This means ASD model can be extended to construct ALM overlay on a large-scale.

After these simulations above, the conclusion can be drawn that both of ASD model and ASD-TS model can construct small-scale and large-scale ALM overlay properly. Their feasibilities are well.

* 1. Performance comparison

After finishing the feasibility analysis, this paper will make a comparison between NICE, a traditional ALM model, and these two models presented in this paper.

In contrast test, the upload bandwidth of each node will be triple as in feasibility analysis, sine NICE protocol requires that node has a relatively high upload bandwidth. The other parameters in this comparison will keep the same as the large-scale simulation in feasibility analysis.

To analysis these AML models, this paper mainly compares two factors: joining delay and delivery delay. The result is shown in fig. 13 and fig. 14

1. Joining delay (s)

The result of joining delay is shown in fig. 13. Joining delay means the lag from a node sending the joining requirement to this node finishing the joining process. In this simulation, 95% nodes in the ASD model finished their joining process in 1.2s, and only 57% nodes in the NICE model did this process in the same time. Furthermore, there were 5% nodes needs more than 3s to finish their joining process in NICE. The result of ASD-TS model is the best, since the Low-timestamp first insert operation makes nodes tend to use insert operation instead of the switch operation and insert operation needs less message exchanging. Finally the average of joining delay was 420ms in ASD model ,251ms in ASD-TS model, and 1304ms in NICE. This result means that ASD and ASD-TS model has an advantage over NICE in joining delay. This may occur because of the structure detection process. The before-hand structure detection makes new node could join the network with less message exchanging.

1. Delivery delay (s)

Delivery delay of a node means the delay between server node sending a data and the data received by the node. From the result shown in fig. 14, the largest delivery delay in ASD model was 802ms and it was 1282ms in NICE. The ASD-TS model has the best performance, 364ms for the largest delivery delay. But when it comes to the average delivery delay, the ASD model has a bad performance. The average delivery delay of ASD model is 477ms and it is 427ms in NICE and 243ms in ASD-TS model. Since ASD model has a bad branch balancing, the performance of delivery delay is not good. But the improved ASD-TS model is good at this factor, since it has better branch balancing and it can use the upload bandwidth of every node efficient.

* 1. Performance details

After the comparison test, this paper will make several statistics to figure out the performance details of ASD-TS model. In this section, the simulation environment is the same as the comparison simulation.

The first statistic is about the delivery delay of each node for every sub-stream.

1. the delivery delay of each node for every sub-stream

In fig. 15 the x-axis means the node number and the y-axis means the delivery delay of each node. To make this figure more readable, the node will be sorted by the delivery delay of stream0 which is the first of all sub-streams. The fig. 15 can be converted to fig. 16.

1. the delivery delay of each node ordered by stream0

In fig. 16, the result shows that if certain node gets one sub-stream with low delay it may get other sub-stream with relatively high delay. This means each node has different priority level in different sub-stream and the model can balance the delivery path in a fair method. The reason of this result is that if one node gets a certain sub-stream early, then it will try to delivery this sub-stream as much as it could do. In the way, it will be the leaf node in the delivery path of other sub-stream. The final result is that every node will get all sub-streams at a near time as shown in the fig. 14.

Except the different priority level, another conclusion of fig. 16 is that stream0 is usually the highest delay of all three sub-streams. This makes the stream0 become the bottleneck of the delivery delay.

To make the analysis more clear, this paper sorts make an accumulation figure of these data. The result is shown in fig.17.

1. delivery delay of each sub-stream

From the fig. 17, it can be conclusion that stream1 is the highest delay of all three sub-streams. This may cause by the primitive grade algorithm. Understanding and solving this problem may be the next research direction.

The second statistic is about the output degree of each node for every sub-stream. From the section 3.1, it can be figured out that the larger output degree of each node for every sub-stream means the better performance the multicast network will be. And the statistic result is shown in fig.18.

1. The output degree statistic for each sub-stream

In the fig. 18, the x-axis means the output degree and the y-axis means the number of nodes whose output degree equals the value of x-axis. So it can be known that the ASD-TS model could make many nodes to delivery one certain sub-stream as many as they can. But it still has a great improved space since there still many nodes deliver the sub-stream at a low degree.

Trying to improve the output degree of each node, this paper tries to increase the threshold which is used to judge whether the structure information is collected enough or not. The final result is shown in fig. 19.

1. the output degree statistic of different threshold

In fig. 19, the x-axis means the output degree and the y-axis means average number of nodes for all three sub-streams. There three results, 'th:20' is the original simulations which means if a node receives 20 structure information messages it will start the joining process. The 'th:500' means a node needs to receive all structure information messages before starting joining process. The average output degrees of each simulation are 4.6 to 'th:20', 6.2 to 'th:100' and 7.6 to 'th:500'. This means the output degree can be improved by collecting more information before starting the joining process. But unfortunately, this action makes the joining delay become larger than original. The results of joining delay of these three simulation are 251ms to 'th:20', 435ms to 'th:100' and 857ms to 'th:500'. And what the worse is that this action cannot improve the performance of delivery delay, since the messages received later are usually responded by the far nodes. These messages have few benefits to improve delivery performance. And the results of the delivery delay are 243ms to 'th:20', 245ms to 'th:100' and 243 to 'th:500'. From the analyses above, it can be known that it is hard to improve the performance of output degree without changing the grade algorithm. So the next research direction may be much more about improving the grade algorithm.

1. Summary

After making an analysis of past ALM models, this paper presents a new ALM overlay constructor method, ASD model, by combining data-driven model and path-driven model. This model uses overlay structure detection to probe the overlay topology and builds delivery path with the topological information. The simulated results show that this model has a good feasibility to construct ALM overlay but has a bad performance at branch balancing. Then this paper presents ASD-TS model to improve this performance. The simulations show that ASD-TS model has a high feasibility and it is good at branch balancing.

On the other hand, from the statistics of the performance detail, it can be conclusion that although the ASD-TS model has some advantage at branch balancing and delivery delay, but it still has much improving space at the output degree of each node. Since the grade algorithm used in this paper is simple and primitive, this needs more research to analysis. And in current grade algorithm, the lag between two nodes just has light effect but it is an important factor in constructing the ALM network actually. So the next research direction may try to solve this problem.

**References:**

1. Banerjee, Suman, Bobby Bhattacharjee, and Christopher Kommareddy. "Scalable application layer multicast." (2002).
2. Tran, Duc A., Kien A. Hua, and Tai Do. "Zigzag: An efficient peer-to-peer scheme for media streaming." INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies. Vol. 2. IEEE, 2003.
3. Chu, Yang-hua, et al. "A case for end system multicast." Selected Areas in Communications, IEEE Journal on 20.8 (2002): 1456-1471.
4. Sripanidkulchai, Kunwadee, et al. "The feasibility of supporting large-scale live streaming applications with dynamic application end-points." ACM SIGCOMM Computer Communication Review. Vol. 34. No. 4. ACM, 2004.
5. Venkataraman, Vidhyashankar, Kaouru Yoshida, and Paul Francis. "Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast." Network Protocols, 2006. ICNP'06. Proceedings of the 2006 14th IEEE International Conference on. IEEE, 2006.
6. Zhang, Xinyan, et al. "CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming." INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE. Vol. 3. IEEE, 2005.
7. Xie, Susu, et al. "Coolstreaming: Design, theory, and practice." Multimedia, IEEE Transactions on 9.8 (2007): 1661-1671.
8. Zhang, Xinyan, Jiangchuan Liu, and Bo Li. "On large-scale peer-to-peer live video distribution: Coolstreaming and its preliminary experimental results." Proc. MMSP. 2005.
9. Li, Bo, et al. "An empirical study of the Coolstreaming+ system." Selected Areas in Communications, IEEE Journal on 25.9 (2007): 1627-1639.
10. Zhang, Meng, et al. "A peer-to-peer network for live media streaming using a push-pull approach." Proceedings of the 13th annual ACM international conference on Multimedia. ACM, 2005.
11. Zhang, Meng, et al. "Understanding the power of pull-based streaming protocol: Can we do better?." Selected Areas in Communications, IEEE Journal on 25.9 (2007): 1678-1694.
12. Zhang, Meng, et al. "Large-scale live media streaming over peer-to-peer networks through global internet." Proceedings of the ACM workshop on Advances in peer-to-peer multimedia streaming. ACM, 2005.
13. Wang, Feng, Yongqiang Xiong, and Jiangchuan Liu. "mtreebone: A hybrid tree/mesh overlay for application-layer live video multicast." Distributed Computing Systems, 2007. ICDCS'07. 27th International Conference on. IEEE, 2007.
14. Wang, Feng, Yongqiang Xiong, and Jiangchuan Liu. "mtreebone: A collaborative tree-mesh overlay network for multicast video streaming." Parallel and Distributed Systems, IEEE Transactions on 21.3 (2010): 379-392.
15. Castro, Miguel, et al. "SplitStream: high-bandwidth multicast in cooperative environments." ACM SIGOPS Operating Systems Review. Vol. 37. No. 5. ACM, 2003.
16. Goyal, Vivek K. "Multiple description coding: Compression meets the network." Signal Processing Magazine, IEEE 18.5 (2001): 74-93.
17. Ingmar Baumgart, Bernhard Heep, Stephan Krause, OverSim: A Flexible Overlay Network Simulation Framework, Proceedings of 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, p. 79-84, Anchorage, AK, USA, May 2007. DOI: 10.1109/GI.2007.4301435.
18. Pongor, György (1993), "OMNeT: Objective Modular Network Testbed", MASCOTS '93: Proceedings of the International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems: 323--326.
19. 彭雪娜, et al. "面向 P2P 流媒体直播的数据分发机制研究进展." 计算机科学 37.005 (2010): 15-20.