Simulation of Active Admission Control Algorithms with OPNET

Bo Rong, Marc Andre Breton, Maria Bennani and Michel Kadoch
Department of Electrical Engineering, Ecole de technologie superieure, University of Quebec
Montreal, Quebec, Canada, H3C 1K3
E-mail: kadoch@ele.etsmtl.ca

Abstract
For future multicast networks, provisioning network resources to meet the QoS requirements is a key issue. The active admission control is a new approach to achieve such resource provisioning. In fact, nowadays active admission control algorithms are becoming more and more popular in congestion control and traffic engineering. In order to study the performance of different active admission control algorithms, employing techniques of simulation is inevitable. The main objective of this paper is to demonstrate how we use OPNET to obtain numerical results and how the results are used to assess the performance of different algorithms.

I. Introduction

Provisioning network resources to meet the QoS requirements is of utmost importance for the development of future networks. Different from some existed mechanisms, we study a new approach of active admission control to achieve such resource provisioning. In fact, active admission control algorithms can be deployed in unicast or multicast networks for variable purposes, such as congestion control and traffic engineering. As an example, in this paper, we study a specific case of employing active admission control to help offer QoS guaranteed service in multicast networks. Since active admission control algorithms are usually complicated, analytical approach is often impotent to analyze and evaluate these algorithms. As a result, OPNET is considered as an inevitable simulation tool to achieve numerical results for these algorithms.

For the QoS guaranteed multicast network, combination of multicast routing and admission control is an appropriate measure to offer better service to real-time multimedia applications. In this paper, we pay attention to the connection setup of multicast applications with stringent QoS requirements. To establish such a connection, both multicast routing and admission control are essential. Multicast routing is to create a multicast delivery tree, and admission control is to check whether or not the application can be admitted on that tree. Once a connection request is accepted, the required resources must be reserved.
In order to provide a multicast delivery tree with QoS guarantee, previous work focused on multicast QoS routing which tries to develop algorithms for finding a cost optimal tree with certain QoS constraints [1,2,3,4,5]. For these algorithms, the admission control is often considered as a by-product of QoS routing and resource reservation. If the routing algorithm can find a route meeting the QoS requirements and the resource reservation is successfully done along the selected route, the connection request is accepted; otherwise, the request is rejected.

Different from existed work, in this paper we intend to integrate multicast routing with active admission control. This way, the admission control is no longer a by-product of routing algorithm, it can affect the routing result. The rest of the paper is organized as follows. Firstly, we discuss the integration of multicast routing and active admission control in Section 2. Then a bandwidth-quota based active admission control algorithm is introduced in Section 3 as a simple example to show how to analyze and evaluate the performance of an algorithm by analytical approach and by OPNET. In Section 4, an adaptable active admission control algorithm that can not be studied by analytical approach is discussed. Moreover, the numerical results of this algorithm obtained by OPNET are carefully analyzed. In the end, Section 5 summarizes our results.

II. Integration of Multicast Routing and Active Admission Control

A. Multicast Routing for Real-Time Connections

A network can be represented by a connected graph $G(V,E)$ with weights associated with edges. In the graph, the nodes stand for communication endpoints, the edges stand for communication links and the weight on each edge $\{W_l, l \in E\}$ represents the cost of that link. A simple network represented by graph is shown in Fig.1.

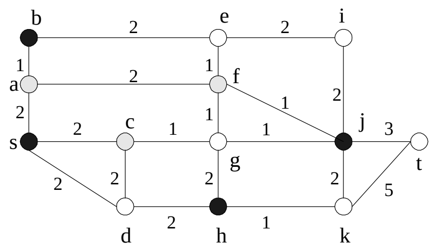

Fig. 1. An example of a network graph

By treating a network as graph, a multicast session can be described as $M=(s,D)$ where s is the source node, $D=\{d_1,\ldots,d_n\}$ is a set of destination nodes. The multicast tree for M is a subtree of $G(V,E)$ rooted from s, which contains all the nodes of D as well as an arbitrary subset of $(V\text{-}D)$, and whose leaf set consists only of a subset of the nodes from D. Along multicast tree, multicast packets should be transmitted simultaneously from s to all the destinations. Parallel transmission can reduce the delay time to send messages to all recipients. Moreover, a minimal number of message copies are transmitted by copying messages only at forks in the multicast tree and it is a good way to reduce the network traffic load.

Let T be a multicast tree of multicast session M, we define the cost of T as follows:

$$COST_T = \sum_{l \in T} W_l \qquad (1)$$

Work supported by Bell Canada and NSERC grants.

$COST_T$ decides the network cost totally used by M. One important aim of multicast routing algorithms is to find out the tree with least cost which can save the consumption of multicast connection greatly. Finding a least cost tree in the weighted graph is regarded as Steiner tree problem[6]. An example of Steiner tree is shown in Fig.2.

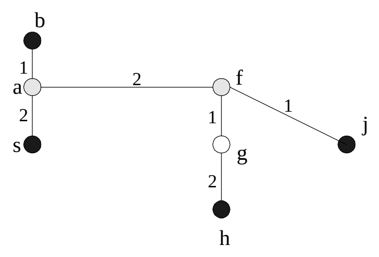

Fig. 2. The Steiner tree for D={b,h,j} in Fig.1

To set up real-time multicast connections, more issues have to be taken into consideration. Most audio or video applications demand quite stringent quality of service to provide smooth play-out at the receiver, thus all these real-time applications usually release a connection request with certain QoS constraints. We can define the end-to-end QoS constraints for the multicast connection as $Q(s, D)$, where s is the source node, D is a set of destination nodes. The QoS requirements usually contain constraints of bandwidth, end-to-end delay, delay jitter and packet loss rate. For the sake of simplicity, only bandwidth and end-to-end delay QoS criteria are addressed in this paper.

B. Active Admission Control in Setup of Real-Time Connections

For the sake of multicast connection setup with QoS guarantee, we use routing and admission control together. There are two kinds of admission control, the passive and the active. The passive admission control uses a policy of FCFS, which means that when a new connection request comes, the network node will do its best to satisfy the QoS requirements of this request without reserving any resource for other connection requests in the future. On the other hand, active admission control employs an active policy, which only allocates part of the resources to new connection request according to the admission control algorithm.

To impose active admission control algorithm on every network node could make the network more complicated and more expensive, therefore the active admission control algorithm should only be deployed on some selected nodes that are important to control the traffic of the whole network. How to choose these active admission control nodes is not very difficult. For example, routers or switches which are in high traffic load and important to control the directions of traffic flows are often selected as active admission control executors. In Fig.1, we choose $\{a,c,f\}$ as active admission control nodes while others as passive admission control nodes.

In general, multicast routing algorithms can be divided to two categories, centralized algorithms or distributed algorithms. We will discuss how to integrate admission control with both of them respectively.

C. Integrating Centralized Multicast Routing with Admission Control

Aiming at integrating centralized multicast routing with admission control in the setup of a real-time multicast connection, we present a three-phase algorithm. Firstly, generate a routing tree based on minimum cost of communication without considering issues of QoS constraints and admission control. Secondly, perform admission test along the generated tree. Thirdly, adjust the multicast tree according to the admission test result. This three-phase algorithm is suited to the network of enough resources and less traffic load. In this case, not a lot of work has to be done in adjusting the originally generated multicast tree.

Because the Steiner tree problem has been proved to be NP-complete, a lot of heuristic algorithms are proposed in recent research, most of them are centralized [7,8,9]. Here, we employ the KMB heuristic [7] as an example to create a sub-optimal multicast tree. The first step of KMB is to construct the complete distance graph G_1 from original network graph G. Nodes in G_1 consist of the source node and all destination nodes in G, and the edges between these nodes represent the shortest path between them. Based on this, an approximate tree can be produced by finding the minimum spanning tree (MST) for the induced graph G_1. However, the expansion of the edges in G_1 into their original shortest paths may produce circles in the MST. These cycles can be easily eliminated by running the minimum spanning tree over the expanded MST. It has been shown that the KMB heuristic finds a tree whose cost is within twice the cost of the corresponding Steiner tree.

After the multicast tree is generated by KMB at the source node without considering QoS constraints, the admission test is performed at both active admission control nodes and passive admission control nodes. Every node has to check out if there are enough resources to guarantee QoS requirements of the request and report the result to the source node.

In order to perform the admission test at each node along the path, we have to divide the end-to-end QoS requirements into local QoS requirements such that a session meeting local QoS requirements at each node will also meet the end-to-end QoS requirements along the path. For bandwidth requirement, it is quite easy, because the local requirement is just same to the end-to-end requirement. But for delay bound, it’s more complicated. Even division policy and proportional division policy are proposed to divide the end-to-end requirements into local requirements in [10].

If there is any failure of admission test, this will require the adjustment of the routing tree formed so far. Two approaches are suggested to solve this problem:

1)QoS Negotiation: Hierarchical coding is employed in the source. When admission test fails in the path to several destination nodes, the QoS negotiation messages with lower QoS parameters that can be guaranteed by the network are sent to the failed destinations. If the destination accepts the new QoS parameters, the path to this destination should not be changed. Otherwise, the destination node should be asked to quit the multicast group, and the subtree related to this destination will be deleted from the original routing tree.

2)Replacement of Unqualified Paths by Rerouting: After knowing the unqualified links and nodes by admission test, we can delete these links and nodes from the network graph and regenerate a new multicast tree, then the admission test will be performed again along the new tree.

There has been already some research on centralized heuristics of delay-bounded multicast routing [1,3]. They aim at finding a multicast tree which minimizes the network cost under the constraint that the delay from the source to any destination is within a bound. For these routing algorithms, we can perform only bandwidth admission test along the multicast tree.

D. Integrating Distributed Multicast Routing with Admission Control

As centralized multicast routing algorithms are difficult to be deployed in large scale networks, some distributed heuristics of finding approximate least-cost multicast tree were proposed in recent research [2,5]. As to these distributed routing algorithms, it seems easier for them to be combined with admission control. The admission test could be done at every node, while the multicast tree is growing from the source to all destinations. In [11], Xiaohua Jia et al. proposed a scheme of integrating distributed multicast routing with passive admission control, which can be easily extended to work also with active admission control. In this scheme, two distributed routing algorithms work together. One is a heuristic mimicking Prim's MST algorithm to compute approximate Steiner tree, another is the shortest path tree (SPT) which has the minimal delay from the source to each destination. Admission test should be performed along two routing trees in parallel. When the Steiner tree heuristic path to a destination fails the admission test, the corresponding SPT path will be included into the final routing tree. If the SPT path fails as well, the requested connection will be rejected. The final routing tree is made out by the Steiner tree heuristic and the SPT together.

In Xiaohua Jia's scheme, the SPT is considered as an algorithm for replacement of unqualified paths produced by Steiner tree heuristic. Moreover, as a supplement to this scheme, when both Steiner tree heuristic path and SPT path fail admission test, QoS negotiation mechanism should be introduced to ask destination nodes to accept a lower QoS level that the routing tree can guarantee.

E. QoS Analysis and Admission Control Policy Choosing

For the reason of simplicity, we only investigate QoS analysis for bandwidth and end-to-end delay in this paper. Compared with delay constraint, bandwidth constraint is easier to be analyzed. Firstly, the local bandwidth constraint for every intermediate node along the path from the source to all destinations is just same to the end-to-end bandwidth constraint. Secondly, it's quite easy to map bandwidth requirement to resource requirements on the network nodes. Meanwhile, the delay constraint of real-time multicast connection is more complex. The delays of a packet at a node usually contain switching delay, queuing delay, transmission delay and propagation delay. The switching delay, transmission delay and propagation delay are constants for a network node and a link. They take much less time than queuing delay when the traffic is busy. The queuing delay is a variable, which depends on the traffic load. In fact, the whole delay on a path is determined by the summary of delays at all nodes and links of that path, hence only guaranteeing the delay at one node is not of great significance. Furthermore, it's not easy to map delay requirement to resource requirements on the network nodes.

As a result of above QoS analysis, we choose passive admission control for end-to-end delay requirement at every network node while use active admission control for bandwidth requirement at active admission control nodes. By employing passive admission control policy for end-to-end delay requirement, a lot of complexity can be avoided and the network architecture is easy to be understood.

III. Comparison of Analytical Approach and OPNET Simulation

A. Bandwidth-Quota Based Active Admission Control Algorithm

In this part, we propose an active admission control algorithm for bandwidth requirement, which can serve as a simple example to show how to analyze and evaluate the performance of an algorithm by both analytical approach and OPNET. When the network is overloaded, on some critical links that can provide broad bandwidth and low transmission delay, we often want to limit the accepting of low-bandwidth connections. Too many low-bandwidth connections on critical links can produce bandwidth fragmentation, which are harmful for these links to accept more high-bandwidth connections. To avoid bandwidth fragmentation, we develop a bandwidth-quota based active admission control algorithm which can give preference to high-bandwidth connections.
Our algorithm is designated to be deployed on the active admission control node to control the access to a critical link of this node. The basic architecture of this algorithm is shown in Fig.3.

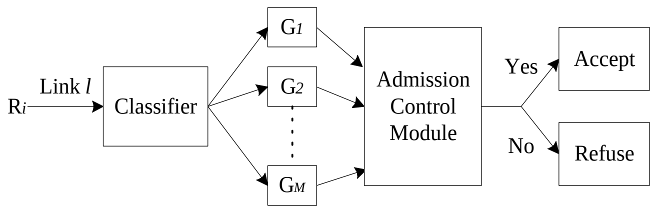

Fig. 3. The basic architecture of our algorithm

In this algorithm, when the i^{th} connection request R_i with bandwidth requirement B_i arrives on link l, the classifier should put it into one of different groups according to the value of B_i. Consider the case where bandwidth requirements on link l can be classified by the values $b_M > \ldots > b_1 > b_0 = 0$ (b_M is the maximum value of B_i permitted on link l), each value corresponding to a QoS level k (k=1,…,M). If $b_{k-1} < B_i \leq b_k$, then the classifier put R_i into group k, which can be denoted by R_i , $B_i \in G_k$. Simply, a request belonging to G_k can be called a G_k request. After carefully choosing parameters such as M and b_k, we consider G_M as the group of highest-bandwidth connections while G_1 as the group of lowest-bandwidth connections. If R_i is accepted by the admission control module, then the corresponding bandwidth should be allocated and R_i begins to transmit data on link l. We define the beginning time and the ending time for R_i to be served on link l as $T_B(i)$ and $T_E(i)$ respectively. Moreover, we say R_i or B_i is alive at time $t(t>0)$, when R_i is accepted by the admission control module and $T_B(i) < t < T_E(i)$.

In order to show preference to high-bandwidth connections, we have to find an algorithm to control the access of link l. To achieve this goal, for any time t and QoS level k (k=1,…,M), we impose upper bound U_k on the sum of bandwidth allocated to all alive connection requests belonging to group G_k. Let C_l be the whole bandwidth capacity of link l, we divide C_l into two parts, presented by $C_l = C_q + C_b$. Here, C_q stands for the bandwidth capacity of QoS guaranteed service controlled by our active admission control algorithm. C_b represents the bandwidth capacity of best-effort service which is designated to support applications without QoS requirements. If C_q is given, our admission control policy can be described as follows:

At any time t, on link l, the following conditions must be satisfied simultaneously:

$$\sum_{\text{All alive } Bi} Bi \le Cq \qquad (2)$$

$$\sum_{\text{All alive } Bi \in G_k} Bi \le U_k, \quad k = 1, \dots, M \qquad (3)$$

Condition (2) indicates that the sum of bandwidth used by all alive connections on link l at time t can not surpass the whole available capacity C_q. Condition (3) indicates that the sum of bandwidth used by all alive connections belonging to G_k can not surpass the upper bound U_k, where k=1,…,M. The value of upper bound U_k decides how much quota you allocate to the G_k requests. In general, if U_k takes a high value, that means the limitation on G_k requests is loose. On the other hand, if U_k takes a low value, that means a strict limitation is imposed on G_k requests. As the group of highest-bandwidth connections, G_M has the highest priority to use bandwidth. Therefore, U_M is always given the same value that C_q holds. Furthermore, the bandwidth requirement of a G_{M-1} request is lower than that of a G_M request, but higher than that of a request from other groups. For this reason, we are inclined to give U_{M-1} a high value as long as it does not damage the acceptance of G_M requests. In addition, more work should be done on how to choose the values of M and b_k according to the traffics in the network. Good parameters are crucial to benefit the performance of the whole network.

B. Simulation by Using Analytical Approach

In this section, we concentrate on how to achieve numerical results by analytical approach to evaluate the performance of our admission control algorithm. We suppose there are three kinds of multicast connection requests in the network: 0.5Mbps for low-bit-rate video applications in group G_1, 1.5Mbps for MPEG-1 applications in group G_2 and 6.0Mbps for MPEG-2 applications in group G_3. Obviously, compared with G_3, G_1 and G_2 are groups of low-bandwidth connections. Moreover, these three kinds of requests are assumed to arrive according to Poisson process independently, and the service times are exponentially distributed. As shown in Fig.4, the system can be described as a Markov chain with state space Ω. Here, Ω represents the set of vector (n_1,n_2,n_3), where n_1, n_2 and n_3 denote the number of alive connection requests on link l belonging to group G_1, G_2 and G_3 respectively. By this model, the blocking probabilities of these three groups can be computed easily.

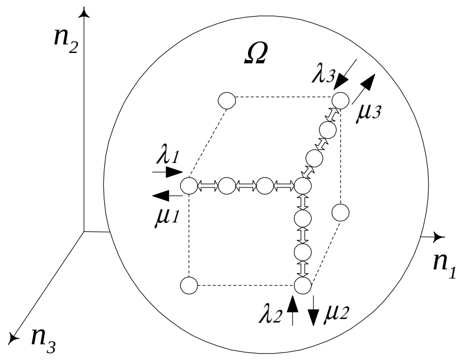

Fig. 4. System model described by a Markov chain with state space Ω

The parameters used in this model are defined below:
C_q(Mbps): bandwidth capacity for QoS guaranteed service on link l.
BG_k(Mbps): bandwidth requirement of a G_k request (k=1,2,3).
U_k(Mbps): upper bound of available bandwidth to G_k requests.
Ω: set of admissible states under a given policy. For our admission control algorithm, Ω is a set of (n_1,n_2,n_3) subject to following conditions:
$n_1, n_2, n_3 \ge 0$ and $0.5n_1+1.5n_2+6n_3 \le C_q$ and $0.5n_1 \le U_1$ and $1.5n_2 \le U_2$;
λ_k(calls/sec): average arrival rate of G_k requests.
$1/\mu_k$(sec/call): average service time for a G_k request.
$\alpha_k = \lambda_k / \mu_k$.
$P(n_1,n_2,n_3)$: probability of n_1 requests in G_1, n_2 requests in G_2 and n_3 requests in G_3.
Pb_k: blocking probability for G_k requests.
$Thrput_k$: throughput for G_k requests.

According to [12], $P(n_1,n_2,n_3)$ has the following product form:

$$P(n_1, n_2, n_3) = \frac{1}{H(\Omega)} \prod_{i=1}^{3} \alpha_i / n_i! \qquad all\ (n_1, n_2, n_3) \in \Omega \qquad (4)$$
$$where \quad H(\Omega) = \sum_{(n_1,n_2,n_3)\in\Omega} \prod_{i=1}^{3} \alpha_i / n_i!$$

The blocking probability for group G_k, denoted by Pb_k, is given by:

$$Pb_k = \sum_{(n_1,n_2,n_3)\in\Omega_k} P(n_1, n_2, n_3) = H(\Omega_k) / H(\Omega) \qquad (5)$$
$$where$$
$$\Omega_k = \{(n_1, n_2, n_3) \mid (n_1, n_2, n_3) \in \Omega \ \ and \ \ (n_1, n_2, n_3) + e_k \notin \Omega\}$$

Here, e_k is a vector from the set {(1,0,0), (0,1,0), (0,0,1)}, where 1 is in the kth position of that vector. Ω_1, Ω_2 and Ω_3 are the sets of blocking states for requests of G_1, G_2 and G_3 respectively. With formula (4) and (5), numerical results with various parameters can be achieved easily.

C. Simulation by Using OPNET Modeler

This section describes the implementation of the simulation for bandwidth-quota based active admission control algorithm using OPNET Modeler. The network model of the implementation contains an active admission control node model to measure the performance of our algorithm. As shown in Fig.5, this node model consists of a few sources (*G1*, *G2* and *G3*), an admission control module and a sink.

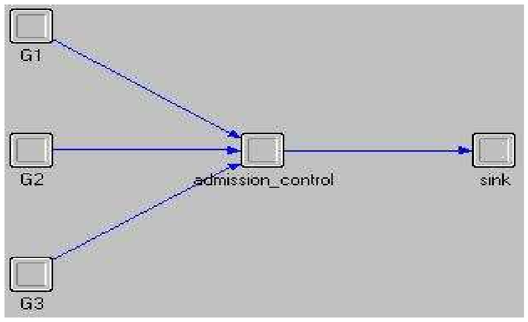

Fig. 5. Node model of active admission control

The basic function of this node model is to mimic the real process offered to a connection request, including admission control, resource allocation and resource reclaim. The sources in Fig.5 are designated to generate the connection requests according to Poisson process by using OPNET's *simple_source* process model and our own packet format. As shown in Fig. 6, our packet format consists of 3 fields (i.e. *groupID*, *rate* and *service time*) which are given different values at different sources for different packets. Here, *groupID* indicates which group the connection request belongs to; *rate* denotes the required bandwidth of this request; *service time* represents how much time this connection should take, which is a random variable of exponential distribution.

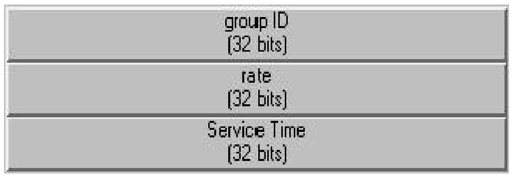

Fig. 6. Packet format

Generated packets from all sources are sent to the *admission_control* module. After receiving a packet, the *admission_control* module extracts the information of a certain connection request from the received packet and executes the admission control algorithm. If the connection request is declined, the packet of this connection request should be sent to the *sink* and deleted. On the contrary, if the connection request is accepted, the *admission_control* module has to reserve corresponding bandwidth resources for it and keep the information of this alive connection before it is out of service time and deleted. In general, it needs three steps to process an accepted connection request. Firstly, once we decide to accept a new connection request, we must increase the number of living connections for that group and decrease the number of available bandwidth. Secondly, we have to calculate the departure time of this specific connection and store it. Thirdly, when the departure time is reached, this connection is not alive anymore. As a result, we have to release the bandwidth resources occupied by this connection and send this connection to the *sink*.

Fig.7 shows the process model used by the *admission_control* module. To facilitate future changes, the code should be optimized to be able to add the specifications of a new group easily. For this reason, in the header file of this process model, we declare two structures which are shown in Table 1. The corresponding arrays of these two structures are also created to manage the information of groups and connections.

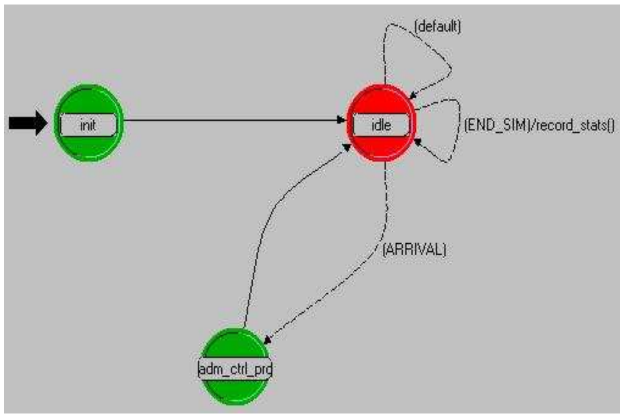

Fig. 7. Process model of *admission_control*

Structure Name	**Variables in the Structure**
group_definition	(Int) on_off
	(Int) group_ID
	(Double) group_rate
	(Double) total_pkt_rcv
	(Double) total_pkt_refused
	(Double) number_alive_connexion
	(Double) number_total_average
group_and_departure_time	(Double) elem
	(Double) dep_time

Table 1: Structure definitions

The process model of *admission_control* begins with executing the *init* state which is used to run the startup initializations before passing the control to the *idle* state. The *idle* state is executed each time when an interruption is generated. Basically, this state is designated to verify if the departure time of a living connection is exceeded. Moreover, the average number of living connections (*number_alive_connexion*) all along the simulation is also calculated in this state. This average number will be used at the end of the simulation to calculate the throughput of a specific group.

When a *stream* interrupt is generated, the control is passed to the *adm_ctrl_process* state which is used to implement the admission control algorithm. The flowchart of the code inside this state is described in Fig. 8. In this figure, the *admission* *decision* module is the most important part, because it implements all the criteria used to make admission decision. For bandwidth-quota based active admission control algorithm, formula (2) and (3) have to be met.

When an *end of simulation* interrupt is generated, a function named *record_stats()* is called to calculate all the numerical results and save the results into a scalar file. This function has been implemented to register the statistics of all groups. In fact, we use a loop to scan the entire array of *group_definition* to find out all the statistical information. We follow the flowchart shown in Fig. 9 to calculate the statistics of all groups. The numerical results usually contain blocking probability, throughput and bandwidth utility efficiency of each group.

Furthermore, our implementation can be easily extended to simulate other active admission control algorithms. If a new algorithm needs to be studied, the major part that has to be changed is the code of *admission decision* module in Fig. 8.

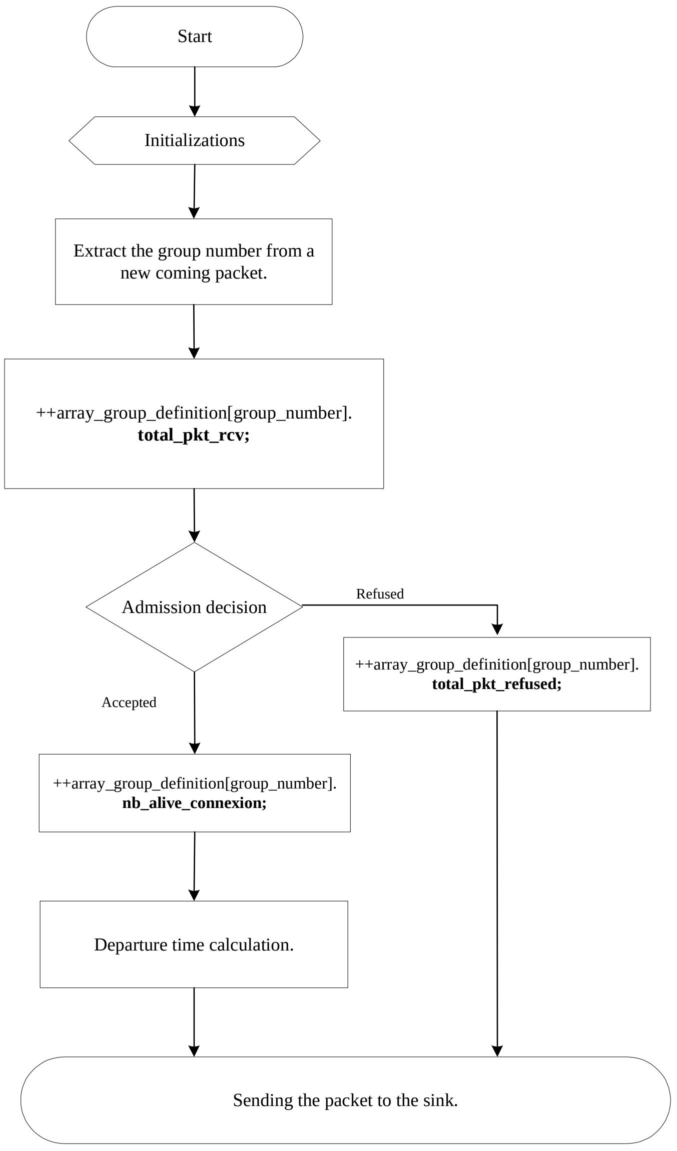
Start
Initializations
Extract the group number from a new coming packet.
++array_group_definition[group_number]. total_pkt_rcv;
Admission decision
Refused
Accepted
++array_group_definition[group_number]. total_pkt_refused;
++array_group_definition[group_number]. nb_alive_connexion;
Departure time calculation.
Sending the packet to the sink.

Fig. 8. Flowchart of the code inside *adm_ctrl_process* state

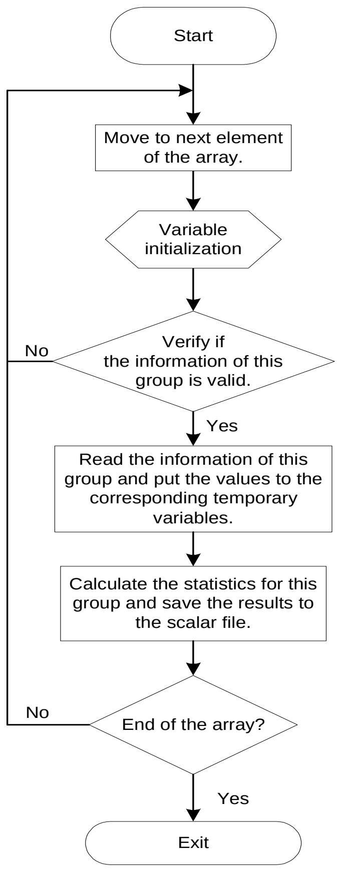
Start
Move to next element of the array.
Variable initialization
Verify if the information of this group is valid.
No
Yes
Read the information of this group and put the values to the corresponding temporary variables.
Calculate the statistics for this group and save the results to the scalar file.
End of the array?
No
Yes
Exit

Fig. 9 Flowchart of the *record_stats()* function

D. Numerical Results

Using both analytical approach and OPNET Modeler, numerical results with various parameters have been calculated and compared. Due to the space limitation of this paper, only results with typical parameters are given out. From Fig.10 to Fig.12, it can be seen clearly that the numerical results from analytical approach and OPNET Modeler are almost same. The first numerical result is shown in Fig.10, demonstrating the effect of varying upper bound of G_1, while other parameters are held constant. The parameters are set as the following:
C_q=155Mbps, α_1=400, α_2=80, α_3=25, U_3=C_q, U_2=50Mbps, U_1= 0~155Mbps;

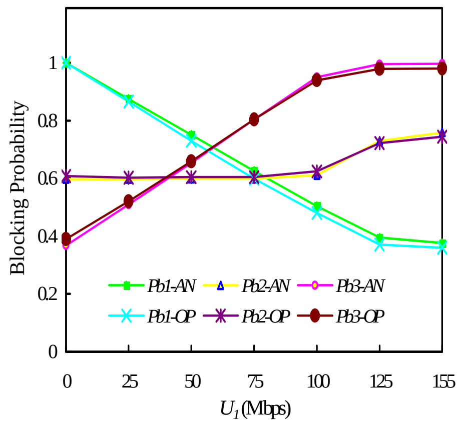

Fig. 10. Blocking probabilities versus upper bound U_1 by both analytical approach (AN) and OPNET (OP)

In Fig.10, while other parameters are kept constant, U_1 varies from 0Mbps to 155Mbps. When U_1 takes a value of 155Mbps, it is the same as having no upper bound on G_1 requests. On the other hand, when U_1 takes a value of 0Mbps, this means that the admission control algorithm does not allow any G_1 request to be accepted in the system. Furthermore, let's analyze Fig.10 with more details. While U_1 is growing from 0Mbps to 155Mbps, the limitation on G_1 is released gradually. As a result, the blocking probability of G_1 is decreased while the blocking probabilities of G_2 and G_3 are increased. This effect is reasonable, because connections belonging to G_1 share more and more resources as its upper bound is going up. However, we can see that the blocking probabilities of G_2 and G_3 are affected differently by G_1. For requests in G_3, the blocking rate is going up dramatically with the increment of U_1. But for G_2, the blocking rate is almost stable before U_1 grows to 100Mbps. In other words, when U_1 is below 100Mbps, the requests in G_1 mainly compete and rob the bandwidth resources from requests in G_3. When U_1 is greater than 100Mbps, G_1 can't obtain more resources by only competing with G_3, therefore it has to rob the bandwidth resources from G_2. The reason of this phenomenon is that requests of lower bandwidth usually have higher priority in competition of bandwidth resources. To prevent this effect from doing harm to high-bandwidth requests, we can choose small values for the upper bounds on low-bandwidth connection groups.

The throughput of each group using the same set of experimental parameters is shown in Fig.11. If the average service times are set to one, the throughput of group G_k can be computed using $Thrput_k=\lambda_k(1\text{-}Pb_k)BG_k$. From Fig.11, the amount of bandwidth resources taken by different groups can be known exactly.

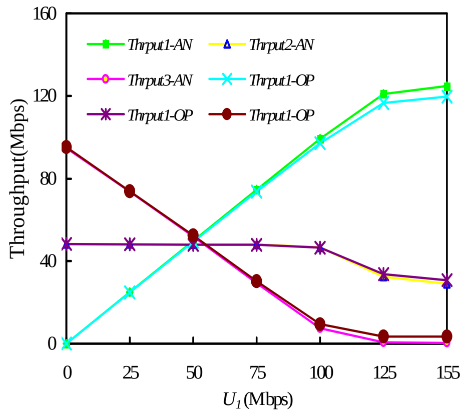

Fig. 11. Throughputs versus upper bound U_1 by both analytical approach (AN) and OPNET (OP)

Numerical result of the second set of parameters is shown in Fig.12, demonstrating the effect of varying the value of α_1, while other parameters are held constant. The parameters are set as the following:

C_q=155Mbps, α_1=50~500, α_2=80, α_3=25, U_3=C_q, U_2=50Mbps, U_1=25Mbps;

From Fig.12, the effect of our algorithm is shown clearly. We choose a low value for the upper bound of G_1 requests (U_1=25Mbps). While α_1 varies from 50 to 500, other parameters are held constant. As α_1 is growing, more and more G_1 requests join the competition for bandwidth resources. Due to the protection from U_1, the blocking probabilities of G_2 and G_3 are almost constant.

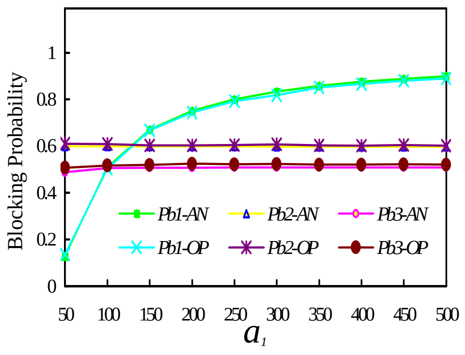

Fig. 12. Blocking probabilities versus parameter α_1 by both analytical approach (AN) and OPNET (OP)

However, the bandwidth-quota based active admission control algorithm can only run well when the arrival rates of all traffics are well known. If the arrival rates of these groups are different from the designated rates, the bandwidth utility efficiency of link l should be damaged. For this reason, we develop another new algorithm in section 4.

IV. Simulation of Adaptable Active Admission Control Algorithm by Using OPNET

A. The Admission Control Algorithm of Dynamic Bandwidth Allocation with Adaptive Constraint

Although analytical approach can be applied to study certain active admission control algorithms, it quickly becomes too complicated and intractable when the source model and the admission control algorithm become complex. In this section, we use OPNET Modeler to study an adaptable active admission control algorithm named dynamic bandwidth allocation with adaptive constraint, which is unable to be investigated by analytical approach according to the theory of [13].

The new algorithm shares the same objective that bandwidth-quota based active admission control algorithm holds. It is developed to protect the acceptance of high-bandwidth connections on a critical link when the network is overloaded. Moreover, compared with bandwidth-quota based active admission control algorithm, the new algorithm also shares the same basic architecture shown in Fig.3, except that the part of *Admission Control Module* is different. However, different from section 3, the bandwidth utility efficiency is taken into account in this section to assess the performance of the new algorithm. The algorithm of dynamic bandwidth allocation with adaptive constraint will be introduced in a step by step way. After discussing some basic admission control algorithms, the new algorithm will be studied as an improvement to these algorithms.

(1) FCFS

FCFS is the simplest admission control policy, which means to accept any new coming connection request regardless of its priority type, as long as there are enough resources to accommodate it.

(2)Fixed Bandwidth Allocation

Under the fixed bandwidth allocation policy, different groups are given different fixed bandwidth resources respectively. When the bandwidth resources allocated to one group is exhausted, a new coming request of this group will be rejected by the server even though there are still available bandwidth resources to other groups. If C_q stands for the bandwidth capacity of QoS guaranteed service on link *l* and BF_k denotes the fixed bandwidth resources allocated to group G_k, the following formula holds:

$$\sum_{k=1}^{M} BF_k = Cq \qquad (6)$$

This algorithm can keep different groups isolated from each other and protect the acceptance rate of high-bandwidth connection groups from being damaged by low-bandwidth connection groups. Nevertheless, if the arrival rates of these groups are different from the expected arrival rates, the bandwidth utility efficiency of link *l* is not very well.
(3)Dynamic Bandwidth Allocation with Adaptive Constraint
Compared with fixed bandwidth allocation policy, besides some fixed bandwidth resources to each group, the additional shared bandwidth resources C_q^{sh} is allocated for all the groups in this algorithm. This shared bandwidth resource policy can be described by the formula below.

$$\sum_{k=1}^{M} BF_k = Cq - C_q^{sh} \qquad (7)$$

Moreover, in order to show preference to high-bandwidth connections, C_q^{sh} has to be shared by all groups under an adaptive constraint policy. Let U_k(t) denote the upper bound of the available bandwidth from C_q^{sh} to G_k requests at time *t* (*t*>0), then the adaptive constraint policy can be defined as the following:

$$U_k(t) = \frac{x\%\left[\sum_{j=k+1}^{M} SLB_{sh}^{j}(t)\right]}{(M-k)} \quad , \quad k = 1,\dots, M-1 \qquad (8)$$

$$U_M(t) = C_q^{sh} \qquad (9)$$

where, $SLB_{sh}^{j}(t)$ is the sum of living bandwidth occupied by group G_j in C_q^{sh} at time *t*, and *x* is the upper bound coefficient.

Due to the introduction of C_q^{sh} and the adaptive constraint, this algorithm can offer more flexibility. Firstly, some fixed bandwidth resources allocated to each group guarantee the basic acceptance of a certain number of connection requests for each group. Secondly, the shared bandwidth resources C_q^{sh} is introduced to increase the bandwidth utility efficiency in case of unexpected arrival rates. Thirdly, in order to give high-bandwidth connection groups higher priority in C_q^{sh} area, the adaptive constraint is added. Furthermore, for the sake of real efficiency, we have to choose appropriate values for each parameter of this algorithm according to the need of a certain network.

B. Numerical Results Obtained by OPNET

In this section, the numerical results are presented to demonstrate the performance of our new admission control algorithm. To study the algorithms discussed in this section by using OPNET, we mimic the implementation of bandwidth-quota based active admission control algorithm mentioned in section 3. The main changes we make are to extend the structures in Table 1, modify the *admission decision* module in Fig.8 and add some new sources in Fig.5. As a highlight of this paper, we only concentrate on the implementation of the algorithm named dynamic bandwidth allocation with adaptive constraint. Table 2 shows the extended structures used to facilitate the implementation. Fig.13 shows the flowchart of *admission decision* module for the algorithm named dynamic bandwidth allocation with adaptive constraint.

Structure Name	Variables in the Structure
group_definition	(Int) on_off
	(Int) group_ID
	(Double) group_rate
	(Int) number_class_by_group
	(Double) BF
	(Double) total_pkt_rcv
	(Double) total_pkt_refused
	(Double) number_alive_connexion_BF
	(Double) number_alive_connexion_CQSH
	(Double) number_total_average
group_and_departure_time	(Double) elem
	(Double) dep_time
	(Int) CQ_type

Table 2: Extended structure definitions

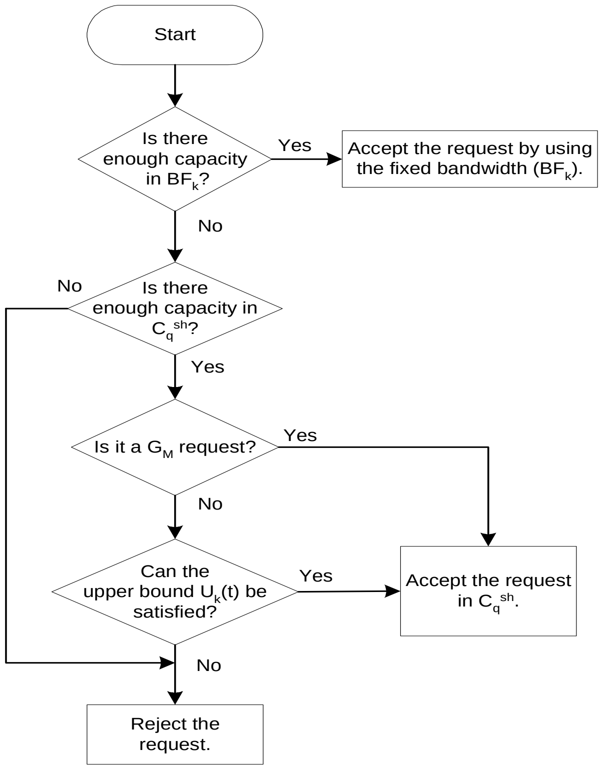
Start
Is there enough capacity in BF$_k$?
Yes
Accept the request by using the fixed bandwidth (BF$_k$).
No
Is there enough capacity in C$_q^{sh}$?
No
Yes
Is it a G$_M$ request?
Yes
No
Can the upper bound U$_k$(t) be satisfied?
Yes
Accept the request in C$_q^{sh}$.
No
Reject the request.

Fig. 13. Flowchart of *admission decision* module for the algorithm named dynamic bandwidth allocation with adaptive constraint

In our simulation, we suppose there are 4 groups of connection requests on link *l*, and in each group there are different bandwidth requirement classes.
(a)group G_1: 56Kbps (class1);
(b)group G_2: 200Kbps(class1), 500Kbps(class2);
(c)group G_3: 1.5Mbps(class1);
(d)group G_4: 4Mbps(class1), 6Mbps(class2).
Obviously, G_1 and G_2 are groups of low-bandwidth connection requests, while G_3 is the medium-bandwidth connection group

and G_4 is high-bandwidth connection group. Moreover, requests of each class in every group are assumed to arrive according to Poisson process independently, and the service times are exponentially distributed.

The parameters used in this model are defined below:
C_q(Mbps)**:** bandwidth capacity for QoS guaranteed service on link l.
C_q^{sh}**:** The shared bandwidth capacity for all groups.
BF_k **:** Fixed bandwidth resources allocated to G_k.
$\lambda_{k,m}$(calls/hour)**:**average arrival rate of class m requests in G_k.
$1/\mu_{k,m}$(hours/call)**:**average service time for class m requests in G_k.
Pb_k**:** blocking probability for G_k requests.
$Thrput_k$: throughput for G_k requests.

With OPNET, numerical results are calculated to show the performance of our algorithm named dynamic bandwidth allocation with adaptive constraint. The first three numerical results are shown in Fig.14, Fig.15 and Fig.16 to demonstrate the blocking probabilities, throughputs and bandwidth utility efficiency of our algorithm under the condition of varying the constraint (upper bound coefficient x) while keeping other parameters constant. In addition, the bandwidth utility efficiency of our algorithm can be computed by using
$Eff=(Thrput_1+Thrput_2+Thrput_3+Thrput_4)/C_q$.
The parameters are set as the following:
C_q=500Mbps,
$\lambda_{1,1}$(56Kbps)=1200(calls/hour), $\lambda_{2,1}$(200Kbps)=600(calls/hour), $\lambda_{2,2}$(500Kbps)=400(calls/hour), $\lambda_{3,1}$(1.5Mbps)=60(calls/hour), $\lambda_{4,1}$(4Mbps)=39(calls/hour), $\lambda_{4,2}$(6Mbps)=16(calls/hour), $1/\mu_{k,m}$=1hour/call (any k or m), C_q^{sh}=250Mbps,
BF_1=5%*(C_q-C_q^{sh}), BF_2=30%*(C_q-C_q^{sh}), BF_3=35%*(C_q-C_q^{sh}), BF_4=30%*(C_q-C_q^{sh}), x=50~300.

From Fig.14 and Fig.15, we can conclude that when upper bound coefficient x takes a small value, the group of high-bandwidth connection requests can be protected very well. On the other hand, Fig.16 shows that when upper bound coefficient x takes a high value, a good bandwidth utility efficiency can be achieved. Therefore, the upper bound coefficient x is crucial to both protecting high-bandwidth connections and improving bandwidth utility efficiency, an appropriate value must be found to compromise these two effects.

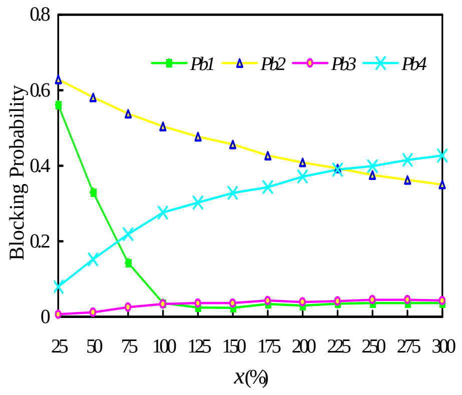

Fig. 14. Blocking probabilities versus upper bound coefficient x (the algorithm of dynamic bandwidth allocation with adaptive constraint)

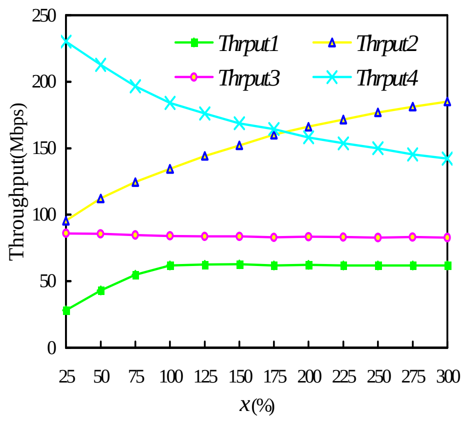

Fig. 15. Throughputs versus upper bound coefficient x (the algorithm of dynamic bandwidth allocation with adaptive constraint)

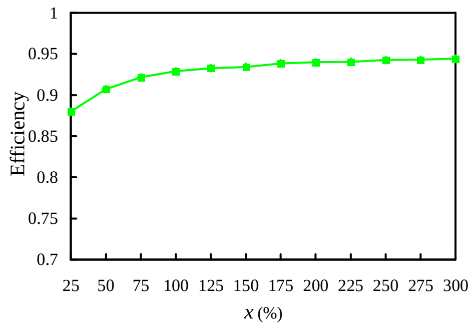

Fig. 16. Bandwidth utility efficiency versus upper bound coefficient x (the algorithm of dynamic bandwidth allocation with adaptive constraint)

More numerical results are shown from Fig.17 to Fig.20. Fig.17, Fig.18 and Fig.19 demonstrate the blocking probabilities obtained by the algorithms of dynamic bandwidth allocation with adaptive constraint, fixed bandwidth allocation and FCFS respectively while Fig.20 shows the efficiencies of all three algorithms. All these four figures try to get the numerical results under the condition of varying the arrival rate of $\lambda_{2,2}$ while other parameters are held constant. The parameters are set as the following:

Common part for all three algorithms:
C_q=500Mbps,
$\lambda_{1,1}$(56Kbps)=1200(calls/hour), $\lambda_{2,1}$(200Kbps)=300(calls/hour), $\lambda_{2,2}$(500Kbps)=100~500 (calls/hour),
$\lambda_{3,1}$(1.5Mbps)=60(calls/hour), $\lambda_{4,1}$(4Mbps)=16(calls/hour), $\lambda_{4,2}$(6Mbps)=12(calls/hour), $1/\mu_{k,m}$=1hour/call (any k or m).

Special part for the algorithm of dynamic bandwidth allocation with adaptive constraint:
C_q^{sh}=250Mbps,
BF_1=5%*(C_q-C_q^{sh}), BF_2=30%*(C_q-C_q^{sh}), BF_3=35%*(C_q-C_q^{sh}), BF_4=30%*(C_q-C_q^{sh}), x=150.

Special part for the algorithm of fixed bandwidth allocation:
BF_1=5%*C_q, BF_2=30%*C_q, BF_3=35%* C_q, BF_4=30%* C_q.

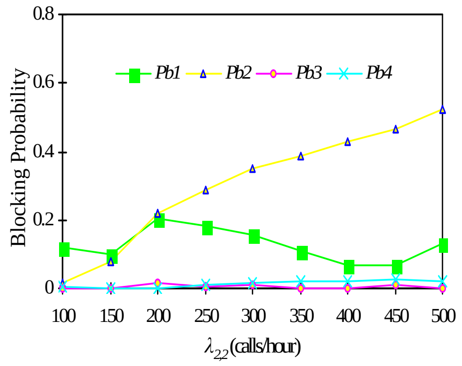

Fig. 17. Blocking probabilities versus arrival rate $\lambda_{2,2}$ (the algorithm of dynamic bandwidth allocation with adaptive constraint)

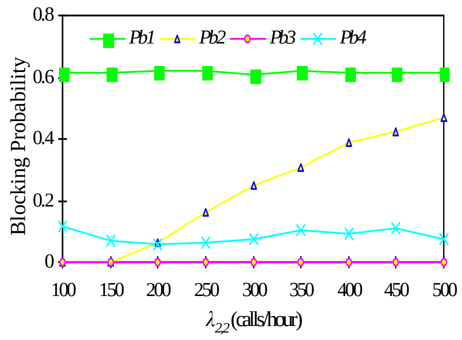

Fig. 18. Blocking probabilities versus arrival rate $\lambda_{2,2}$ (the algorithm of fixed bandwidth allocation)

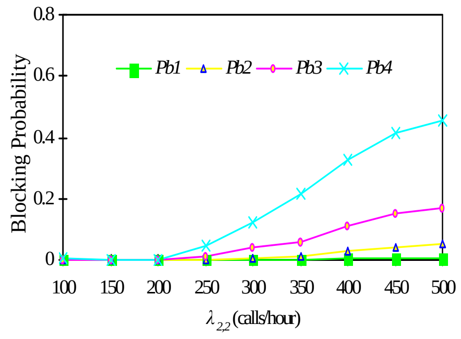

Fig. 19. Blocking probabilities versus arrival rate $\lambda_{2,2}$ (FCFS)

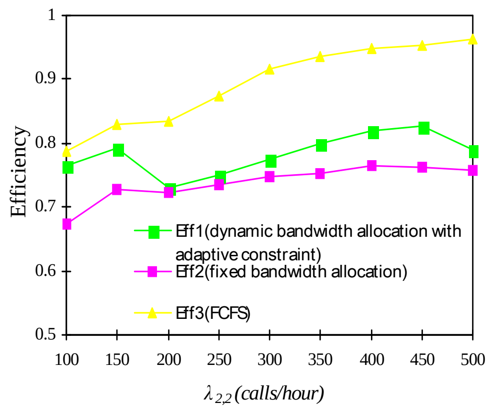

Fig. 20. Bandwidth utility efficiencies versus arrival rate $\lambda_{2,2}$ (the results of all three algorithms)

From Fig.17 to Fig.20, it can be seen clearly that the algorithm of dynamic bandwidth allocation with adaptive constraint has the best performance among all three algorithms. Although FCFS has the highest bandwidth utility efficiency, it can not offer any protection for high-bandwidth connection requests. The algorithm of fixed bandwidth allocation can isolate different groups of requests very well, but its bandwidth utility efficiency is low. The algorithm of dynamic bandwidth allocation with adaptive constraint not only can show the preference to high-bandwidth connection requests, but also has higher bandwidth utility efficiency than the fixed bandwidth allocation algorithm has.

V. Conclusions and Future Work

We have presented a new approach of integrating multicast routing with active admission control. Furthermore, a bandwidth-quota based admission control algorithm is introduced as a simple example to show how to study the performance of active admission control algorithm by using analytical approach and OPNET Modeler respectively. Later on, an adaptable active admission control algorithm named dynamic bandwidth allocation with adaptive constraint is proposed to avoid bandwidth fragmentation and improve the bandwidth utility efficiency. This algorithm is unable to be investigated by analytical approach, therefore we use OPNET Modeler to obtain the numerical results and assess the performance. From the numerical results, we can conclude that our algorithm can ensure the preference for high-bandwidth connections in overload condition while achieving good bandwidth utility efficiency. In the near future, we plan to integrate our active admission control algorithms into the routers modeled by OPNET. By this way, the simulation of a large scale IP multicast network will be developed on OPNET platform to find out the achievements of our algorithms in respect to congestion control and traffic engineering.

References

[1] V. P. Kompella, J. C. Pasquale, and G. C. Polyzo, "Multicast routing for multimedia communication," *IEEE/ACM Transactions on Networking*, pages 286-292, June 1993.

[2] V. P. Kompella, J. C. Pasquale, and G. C. Polyzo, "Two distributed algorithms for multicasting multimedia information," *Proceedings of ICCCN'93*, pages 343-349, 1993.

[3] Q. Zhu, M. Parsa, and J. J. Garcia-Luna-Aceves, "A source-based algorithm for delay-constrained minimum-cost multicasting," *Proc. of IEEE INFOCOM'95*, pages 377-385, 1995.

[4] G. N. Rouskas and I. Baldine, "Multicast routing with end-to-end delay and delay variation constraints," *IEEE Journal on Selected Areas in Communications*, pages 346-356, April 1997.

[5] X. Jia, "A distributed algorithm of delay bounded multicast routing for multimedia applications in wide area networks," *IEEE/ACM Transactions on Networking*, pages 828-837, December 1998.

[6] Gilbert and H. O. Pollack, "Steiner minimal tree," SIAM J. Appl. Math., Vol.16, 1968.

[7] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," *Acta Informatica*, pages 141-145, 1981.

[8] H. Takahashi and A. Matsuyama, “An approximate solution for the Steiner problem in graphs,” *Math. Japonica*, pages 573-577, 1980.

[9] N. F. Maxemchuk, “Video distribution on multicast networks,” *IEEE Journal on Selected Areas in Communications*, pages 357-372, April 1997.

[10] V. Firoiu and D. Towsley, “Call admission and resource reservation for multicast sessions,” *Proc. of IEEE INFOCOM’96*, pages 94-101, 1996.

[11] X. Jia, Y. Zhang, N. Pissinou and K. Makki, “An efficient admission control method of real-time multicast connections in wide area networks,” In *Proc. 7th International Conference on Computer Communications and Networks*, pages 865 –872, 1998.

[12] J. S. Kaufman, “Blocking in a shared resource environment,” *IEEE Transactions on Communications*, pages 1474-1481, October 1981.

[13] J. M. Aein, “A multi-user-class, blocked-calls-cleared demand access mode,” *IEEE Transactions on Communications*, vol. COM-26, Mar. 1978