Evaluation of Lightweight Preemption Algorithms for UPnP QoS Architecture

Lukasz Brewka, Henrik Wessing and Lars Dittmann
Department of Photonics Engineering, Technical University of Denmark, 2800 Lyngby, Denmark
Phone:(+45) 45253620, Email: {ljbr, hewe, ladit}@fotonik.dtu.dk

***Abstract*—Preemption is a common approach for QoS provisioning in multiservice and multiuser networks where the network resources are exhausted. This paper focuses on the analysis of the lightweight algorithms for the UPnP QoS Architecture where the decision about preemption is made by computation power limited home gateway. Based on the data obtained, the comparison of the proposed algorithms shows that the performance of simple methods can give satisfying rejection and preemption rates.**

I. Introduction

The number and functionality of QoS mechanisms available in home networks is becoming a more and more significant factor for design of services and architectures of future networks in an Intelligent Home. This is mainly caused by the growing number of flows in home networks and diversity of traffic types with a clear differentiation of the importance of particular traffic flows. There is also a great focus on managing the home environment with a system that can handle the dynamic character of the network, with devices leaving and joining the network frequently. Service based platforms are a common choice for organizing and controlling the described networks. There is a number of protocols designed for dynamic service discovery that can be used for establishment of home network; UPnP [1], DPWS [2], Bonjour [3], Jini [4] and IGRS [5]. While Bonjour does not explicitly consider QoS, Jini and IGRS are more focused on the end-devices' resources than network's resources, UPnP and DPWS are explicitly describing network QoS mechanisms. UPnP together with its QoS Architecture specification provides a good environment for evaluation of preemption procedures. That is why in the remaining part of this paper we will consider only UPnP QoS Architecture [6] and the analysis of preemption algorithms will be based on the UPnP signaling model. However analysis made here are generic enough that they could be used in any QoS architecture where preemption is possible and managing entities are capable of determining policies of existing and incoming QoS requests.

Preemption is a procedure that allows the admission of the new traffic flow even in a case of insufficient amount of free resources. When the managing entity decides that the arriving traffic is more important than one (or a group) of the flows already occupying the resources, it can release these resources and at the same time declines the previously established QoS (usually equivalent of degrading the QoS to Best Effort). The preemption algorithms described in the literature [7], [8], [9] are aimed on the optimal (or suboptimal) solutions, minimizing rerouting, number of preempted flows and their priority. When centralized preemption is considered authors of [8] showed that the problem is NP-complete. In home networking not all of listed above parameters are of high importance. E.g. rerouting usually is not possible as home network topology is usually quite simple and there will not be many alternative routes, actually the topology is so simple that it is reasonable to make per interface decisions. When it comes to traffic priority and number of preempted flows these are parameters of more importance and they will be studied in this paper in more details. Some studies like [10] consider a random selection algorithms showing that the optimal and suboptimal algorithms are outperforming the random selection but sometimes the latter archives comparable results with much lower complexity. It is also important to mention that limited processing power in the home network justifies the focus on lightweight algorithms with low computing complexity and implementation effort. In this paper we propose three lightweight preemption algorithms that are designed to match general home network topology and processing power capabilities, at the same time being compatible with UPnP QoS Architecture.

The remainder of this paper is organized as follows. Section II is treating the preemption algorithms, section III describes developed UPnP model. This is followed by section IV with the simulation results and their discussion. Finally, the conclusions are placed in section V.

II. UPnP Preemption Study

Preemption is one of the QoS mechanisms available in the UPnP QoS Architecture [6]. Upon failure of the resource reservation procedure, in UPnP's parametrized QoS setup, QoS Manager (QM - the service that is in control of networks QoS negotiation and establishment) can re-attempt the reservation's admittance. This takes place only if the Control Point (CP - entity possessing knowledge about source destination and traffic specification) requested usage of the preemption functionality. If that is a case, QM will request a list of blocking flows' traffic policies, and based on that it will decide about the release of some of the resources. UPnP defines the QM's release command that passes the *Traffic Handle* parameter while calling *ReleaseTrafficQos* action [11]. The protocol does not specify the method for releasing multiple flows. A case

CHINACOM 2010, August 25-27, Beijing, China

DOI 10.4108/chinacom.2010.60

of multiple flows release requires multiple *ReleaseTrafficQos* messages to be sent. The specification also leaves to the implementers the decision procedure, which determines what should be preempted and in what circumstances. Below we present some preemption algorithms that could be used by QM to select the reservations to release.

Proposed Algorithms

In this section we present proposed preemption algorithms and the motivation for their use. The main goal is to propose algorithms with low complexity, that should ease the design and implementation of home network management units like Residential Gateway, settop boxes etc. Obtained results will show whether using simple algorithms provides acceptable results when (a) existing reservation preemption and (b) new reservation rejection rates are considered for particular traffic priorities.

1) First Fit is the simplest algorithm that aims on identification of a single flow which preemption could allow an acceptance of a newly arriving flow. During the search of a particular flow QM scans through the *ExtendedQoSState* (a message received from a QoS Device (QD) upon reservation failure) and once the candidate flow (the flow that consumes sufficient resources and has priority lower than the new flow requesting the QoS) is identified the search is stopped and the release action of the selected candidate flow will be performed. If none of the existing reservations comply with both conditions, the preemption can not take place.

2) Minimal Single Fit algorithm is looking for a single flow, which released frees enough resources that allow setting up QoS for a new flow. Selected for preemption reservation has to be of lower priority than incoming reservation and at the same time, is of the minimum priority out of the existing reservations. The managing unit searches the whole *ExtendedQoSState* message to identify the single flow that could be subjected to preemption. In case of this algorithm, similarly like for *First Fit*, preemption is not possible if there is no single flow, which satisfies both the priority and resources amount conditions.

An obvious disadvantage of described earlier algorithms is that they only look for a single flow reservation that consumes resources required for a newly arriving request. It can often be a case that in situation that there is not single flow that consumes resources that should be given to higher priority reservation, there is a group of lower priority reservations that consist of a number of flows that together are using sufficient amount of resources. This case requires multiple preemption messages to be sent to a single device (due to described earlier, single flow release message in UPnP). This procedure takes more time but can be beneficial in regards to preemption and rejection rates. That is why we introduce the third algorithm.

3) Minimal Group Fit algorithm is looking for a group of flows. All the reservations in this group should be of lower importance comparing to the new reservation and the sum of the resources that they occupy, once released, should allow a new reservation to be successfully performed. The algorithm

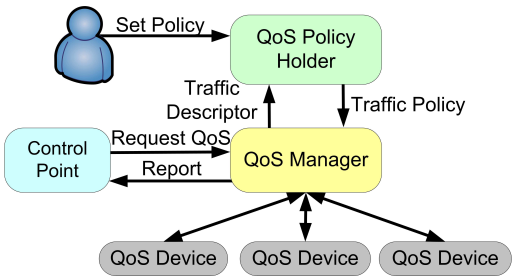

Fig. 1. UPnP architecture

we defined here takes flow's priority as the major deciding factor. The procedure progresses as follows; minimal priority flow is determined and its resources are added to a free-resources variable, if that does not give sufficient resources a second minimal priority flow is identified, its resources are added to a free-resources variable - this is continued until enough resources are freed. The algorithm considers only flows with priority lower than new request's priority and before hand verifies if the procedure described earlier can be completed. Increasing the probability of freeing enough resources for a new reservation by preemption of multiple flows is archived on expense of higher algorithm complexity namely $O(n^2)$ instead of $O(n)$ for *First Fit* and *Minimal Single Fit*. Where n is the number of flows that are occupying the queue at the moment of traffic admission.

III. UPnP Model

The model developed for simulations of the UPnP QoS Architecture is presented in Fig. 1. We developed the CP service that in random exponentially distributed intervals generates the reservation requests with random uniformly distributed priority and resources amount (priority between 0 and 9; resource amount between 5 and 35% of the queue size). The average flow holding time is 120 seconds. The QoS Policy Holder (QPH) is managing the policies and returns requested policy or list of policies to the requesting QM. List of policies is returned for the request containing multiple *Traffic Handles* in a single policy request (typically used during preemption where there are multiple candidates for release - the group of blocking flows). During the simulations we used three QoS Devices of identical structure. Each of the devices is managing its resources by modeling ten queues of different priorities. Once a request for a new reservation arrives, the QD verifies if it is possible to accommodate this reservation in the queue, meaning if:

$$\sum_{ID=1}^{n} Res_{ID} + Res_{new} \leq Res_{total} \tag{1}$$

Where Res_{ID} are the resources occupied by established earlier flows with particular ID, Res_{new} are the resources requested for new flow, and Res_{total} is the total size of each of ten modeled queues.

If the device's queue state allows the admittance of the new flow, the information regarding the flow (ID and resources)

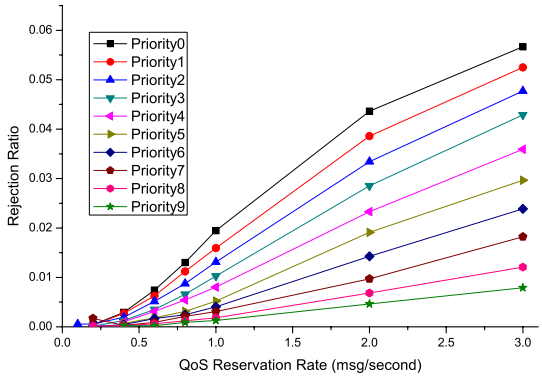

Fig. 2. Rejection ratio for different priority flows as a function of the flow reservation rate for First Fit preemption algorithm

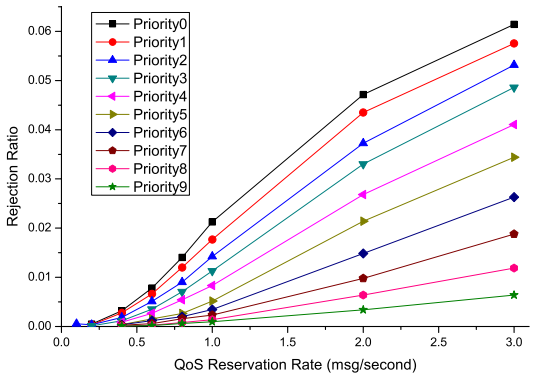

Fig. 3. Rejection ratio for different priority flows as a function of the flow reservation rate for Minimal Single Fit preemption algorithm

are added to a proper queue state and QM is notified about the successfully admitted request. If it is impossible for the device to accommodate the new traffic it will send a fail notification to the QoS Manager. In these circumstances QM will proceed with preemption. First, the QM retrieves the *ExtendedQoSState* from the devices that reservation failed on. The *ExtendedQoSState* message in our implementation contains information of all ten queues together with flow IDs and the resources they occupy. Later QM makes the decision if there is anything that could be preempted in order to make a new reservation possible.

The model used for the work presented here is developed in the OPNET modeling tool [12].

IV. Simulation Results

In this section we present the results of the performed simulations. During the simulations a number of characteristics and measurements were obtained. The performance of different algorithms is evaluated based on; the reservation rejection rates in different classes, the preemption rates for different classes, the exceeding resources preemption, and the average class reservation level.

Fig. 2, 3 and 4 show the rejection rate (measured as the number of rejected notifications for a particular priority over the total number of notifications received; rejections and acceptances) for different preemption algorithms. The simulation results clearly show that all the algorithms provide expected rejection fairness for different classes. The improvement in the rejection ratio of new reservations that are requested by CP is relatively low, especially for the middle priority reservations. Obviously, the algorithms that choose minimal priority flows (*Minimal Single Fit*, *Minimal Group Fit*) for preemption have a higher preemption ratios for those reservations. On the other hand those algorithms protect the highest priority reservations much better, for which the improvement rate is very noticeable reaching the factor of 100 for the highest priority.

To expose more clearly the differences between the rejection rates for different algorithms in Fig. 5 we present how the rejection rate differs with changing reservation generation rate for three chosen classes. This graph depicts well that

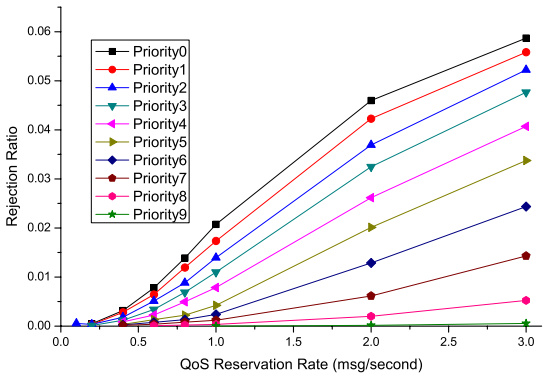

Fig. 4. Rejection ratio for different priority flows as a function of the flow reservation rate for Minimal Group Fit preemption algorithm

protection of the high priority reservations is simply archived by higher rejection ratio for lower priority classes.

Additionally, in Fig. 6 we present how for the proposed algorithms the rejection rates differ with changing priority, for chosen reservation message generation rates (0.6, 1, 2 msg/second). This graph shows that priority class four is the reverse point of the rejection rates for reservation messages

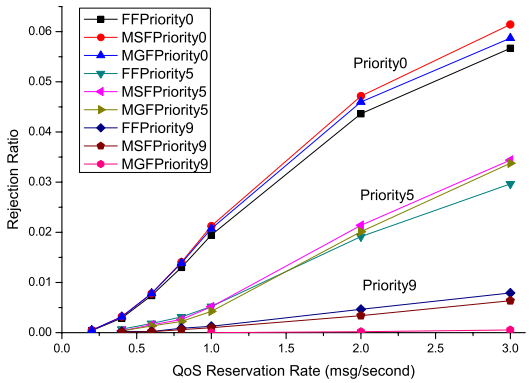

Fig. 5. Rejection ratio for different preemption algorithms and chosen priorities as a function of the flow reservation generation rate

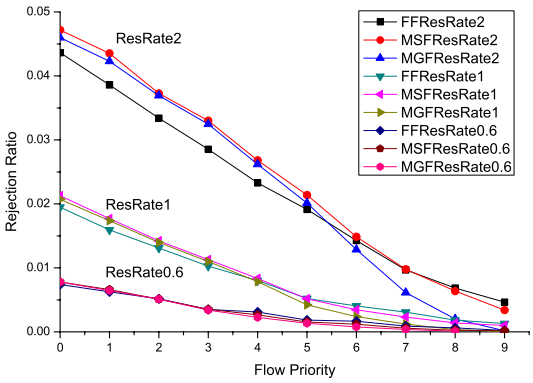

Fig. 6. Rejection ratio for different preemption algorithms as a function of the flow priority for reservation generation rates 0.6, 1 and 2 msg/second

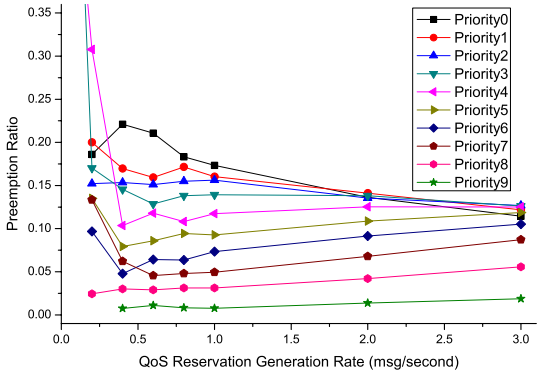

Fig. 7. Preemption ratio for different priority flows as a function of the flow reservation rate for First Fit preemption algorithm

generation ratio 1 msg/second. That means that for *Minimal Single Fit* and *Minimal Group Fit*, comparing to *First Fit*, reservations from classes 5 to 9 are better protected, while reservations from classes 0 to 4 will be more often subjected to rejections. For the reservation request rates 0.6 and 2 (msg/second) the reverse points are moved in direction of respectively lower and higher priorities. That is caused by the fact that for higher QoS request rates there is a higher probability of many high priority reservations occupying the resources (since those reservations will be less often pre-empted) in the time of new reservation arrival, which shifts the average queued priority to higher values.

The results of the preemption study show the amount of reservations preempted in particular classes (note: preemption in the highest class is possible as there are multiple priority levels within the class - depending on the flow ID). Fig. 7, 8 and 9 present the preemption of reservations within different classes for three proposed algorithms. It is clearly visible that the *First Fit* (Fig. 7) algorithm does not create the same degree of separation between preemption levels for different classes. This differentiation is more visible for the *Minimal Single Fit* and *Minimal Group Fit* algorithms (Fig. 8 and 9). Additionally, the *Minimal Group Fit* algorithm lowers the preemption of the highest priority flows even more than the *Minimal Single Fit* algorithm. The data collected for very low reservation rates (0.1 and 0.2) are normalized over small number of releases and as such they are presented only to indicate the preemption tendency. In Fig. 10 we present how the preemption rates change with flow priority for proposed algorithms (data are obtained for reservation message generation rate - 1 msg/second). Also here aforementioned reverse point is visible showing that *First Fit* algorithm is not protecting high priority reservations as well as *Minimal Single Fit* and *Minimal Group Fit* do.

Fig. 11 shows the comparison of the algorithms considering the amount of exceeding bandwidth (BW) that is released during the preemption. The graph shows that both *Minimal Single Fit* and *Minimal Group Fit* are outperformed by *First Fit* algorithm. The difference between the exceeding band-

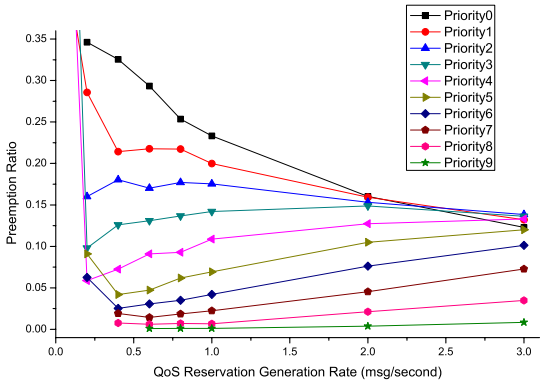

Fig. 8. Preemption ratio for different priority flows as a function of the flow reservation rate for Minimal Single Fit preemption algorithm

width released using described algorithms grows with growing reservation rate. All algorithms perform better in regards to exceeding bandwidth release with growing reservation message rate, which can be explained by higher probability of more small reservations being stored in the queues.

Another performance assessment parameter we analyzed, is the queue utilization obtained using different algorithms. Fig. 12 shows that all the algorithms archive similar queue

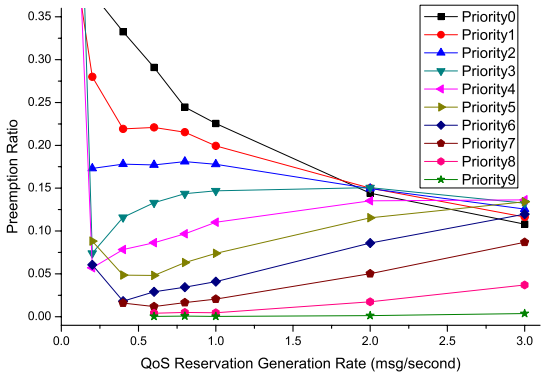

Fig. 9. Preemption ratio for different priority flows as a function of the flow reservation rate for Minimal Group Fit preemption algorithm

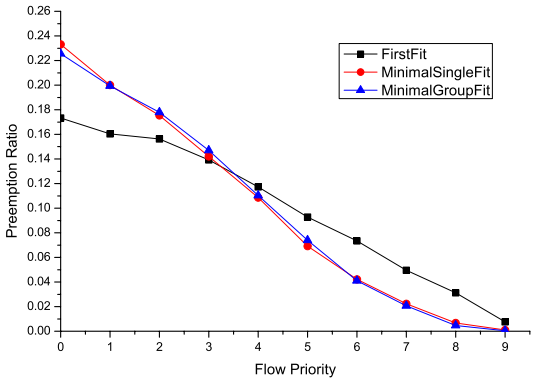

Fig. 10. Preemption ratio for different preemption algorithms as a function of the flow priority for reservation rate 1 msg/second

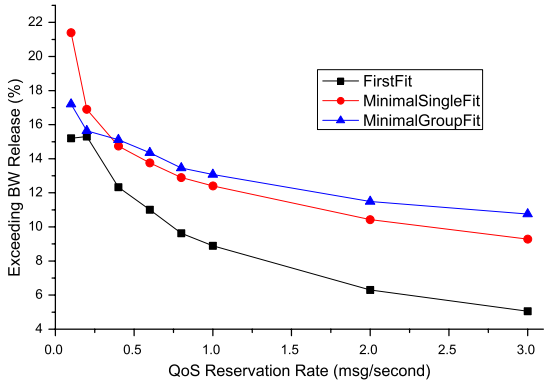

Fig. 11. Exceeding bandwidth released for different algorithms

occupancy level. This means that on the average the same bandwidth is accommodated on devices' interfaces for all the cases. The *Minimal Single Fit* and *Minimal Group Fit* algorithms simply allow more high priority traffic instead of lower priority flows. Though, what was showed before, *First Fit* releases less exceeding bandwidth, the resources used on particular interfaces are on the same level for all three algorithms.

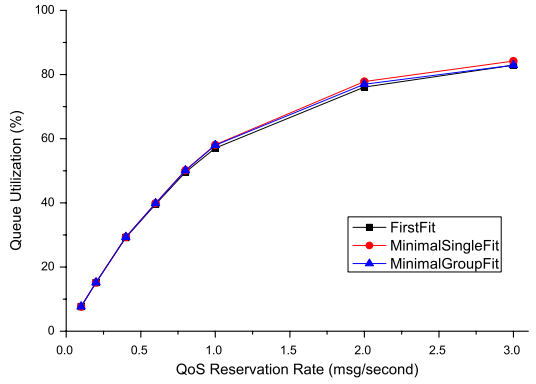

Fig. 12. Queue utilization for different algorithms

V. Conclusions

The work we presented here is an extension and completion of the UPnP QoS Architecture. We have proposed and shown the performance of three algorithms that can be used for preemption in a home network environment under dynamic conditions. The results obtained during algorithms analysis show that even the simplest algorithm proposed i.e. *First Fit* is providing good fairness of both request rejection and reservation preemption. Additionally, *First Fit* performs better than the other proposed algorithms when exceeding bandwidth release is considered. The data we obtained also show that when the highest priority reservations are considered the *Minimal Single Fit* and *Minimal Group Fit* provide much higher level of protection of the highest priority traffic - where *Minimal Group Fit* performs very well. On the other hand one have to be aware of higher complexity ($O(n^2)$ for *Minimal Group Fit* comparing to $O(n)$ for *First Fit* and *Minimal Single Fit*) and the possibility of need for multiple preemptions in order to accommodate a single reservation. Due to good performance and only single flow preemption the studies here indicate that the *Minimal Single Fit* algorithm can be seen as the most suitable for use in the UPnP QoS Architecture. Additionally the advantage of *Minimal Single Fit* over *Minimal Group Fit* is its lower computational complexity.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7) under project 212 352 ALPHA "Architectures for fLexible Photonic Home and Access networks".

References

[1] "Upnp technology the simple, seamless home network a white paper," December 2006.
[2] *Devices Profile for Web Services Version 1.1*, OASIS, http://docs.oasis-open.org/ws-dd/dpws/wsdd-dpws-1.1-spec.html, July 2009.
[3] *Bonjour Overview*, Apple Inc., http://developer.apple.com/ documentation /Cocoa/Conceptual/NetServices/NetServices.pdf, May 2006.
[4] *Jini Architecture Specification*, http://www.jini.org/wiki/Jini_ Architecture_Specification, March 2007.
[5] *Internet Grouping and Resource Sharing*, IGRS Information Industry Association, http://www.igrs.org/en/index/index.asp.
[6] *UPnP QoS Architecture:3 Service Template Version 1.01 For UPnP Version 1.0*, UPnP Forum, November 2008.
[7] S. Jeon, R. T. Abler, and A. E. Goulart, "The optimal connection preemprion algotirhm in a multi-class network," in *IEEE International Conference on Communications, 2002. ICC 2002.*, vol. 4, 2002, pp. 2294–2298.
[8] J. Garay and I. Gopal, "Call preemption in communication networks," in *INFOCOM '92. Eleventh Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE*, vol. 3, New Delhi, India, October 1992, pp. 1043–1050.
[9] T. Shan and O. W. Yang, "Bandwidth management for supporting differentiated-service-aware traffic engineering," in *IEEE International Conference on Communications, 2002. ICC 2002.*, vol. 2, September 2002, pp. 1305 – 1309.
[10] V. Stanisic and M. Devetsikiotis, "A dynamic study of providing quality of service using preemption policies with random selection," in *IEEE International Conference on Communications, 2003. ICC 2003.*, vol. 3, May 2003, pp. 1543–1546.
[11] *UPnP QosManager:3 Service Template Version 1.01 For UPnP Version 1.0*, UPnP Forum, November 2008.
[12] *OPNET Modeler Version 14.5.A*, http://www.opnet.com, March 2007.