QoS Aware Multicast Routing Protocols Evaluation through Simulation

António Costa
Departamento de Informática
Universidade do Minho
Braga, Portugal
costa@di.uminho.pt

Maria João Nicolau
Departamento de Sistemas de Informação
Universidade do Minho
Guimarães, Portugal
joao@dsi.uminho.pt

Joaquim Macedo
Departamento de Informática
Universidade do Minho
Braga, Portugal
macedo@di.uminho.pt

Alexandre Santos
Departamento de Informática
Universidade do Minho
Braga, Portugal
alex@di.uminho.pt

ABSTRACT

In networking research, the simulation is often the single way to overcome the lack of equipment needed for laboratory setup of complex experimental topologies, with diverse traffic pattern scenarios. Even for simpler topologies and traffic scenarios, the simulation remains attractive due to the available facilities on data collection, graphics generation and step by step analysis of different protocol machines.

Inter-domain multicast scenarios, where heterogeneous QoS requirements should be considered, is a particular example where both complexity and resources availability justify the use of simulation.

This paper reports the experience gained by the usage of simulation tools in multicast routing with QoS. It is not focused on the real simulation results reported but on the process used to obtain them. NS-2 has been used as the base of this work.

Keywords

NS2, Multicast Routing, Quality of Service, Simulation Tools

1. INTRODUCTION

Multicast communication is needed to support many applications, in particular applications with special requirements on multimedia flows transfer. Even with good application level multicast solutions, a good multicast routing strategy enables the minimisation of communication cost and the optimisation of the network resources usage. The network layer approach is suitable for inter-domain multicast, where there are relevant scalability and policy dependent unsolved problems. These non local or inter-domain restrictions constraint the set of acceptable solutions.

IP based networks are faced with new services models and a new vocation to become a multi-services universal network. The existing long term research and developments towards a QoS aware IP network, replacing the traditional best-effort network introduced a set of mechanisms inside and outside the network routers. But it is at inter-domain routing level where more efforts and solutions are missing. This is notorious for multicast routing, where applications are sensible to the network available QoS like losses, delays and bandwidth. Another important issue at large scale multicast applications is the existence of users with heterogeneous QoS requirements.

Most of applications needing multicast with QoS are only attractive for large scale contexts. A typical example is the events diffusion, with low commercial impact for traditional mass media and occurring at a large geographic distance from potential interested people.

The goal of this work is to show the usage of a simulation tool as an evaluation framework for inter-domain multicast routing protocols proposals. The implementation of multicast routing protocols in the simulator and the simulation results show that the simulation is an important and useful technique in the study and development of this type of protocols. Simulation results enable a quantitative performance evaluation using different metrics for several traffic patterns and network topologies.

2. NETWORK SIMULATORS

In networking research, the simulation is often the single way to overcome the lack of equipment for complex topologies experimental laboratory setup, with diverse traffic pattern scenarios. Even for simpler topologies and traffic scenarios, the simulation remains attractive due to the available facilities on data collection and graphics generation and step by step analysis of the different protocol machines.

Discrete event simulation techniques are suitable for computer networks modulation and analysis. Data packet flow is seen as a sequence of events occurring at different time instants. Computer networks simulation is a well defined process including a set of methods to describe the network topology, network elements behaviour, to observe network status and an engine to manage and process the events.

There are several commercial and open source computer network simulation packages. Only the more popular will have here a brief description. For a more exhaustive list, please see [9].

Network Simulator 2 (NS-2)[3] is a second version of the simulator developed by Virtual InterNetwork Testbed[8] (VINT). Now, NSF is funding a third version of Network Simulator (NS-3). NS-2 is a very popular discrete event simulator within networking research community. It includes a large number of network protocol models. The NAM (Network Animator) enables the graphic visualisation and protocol checking of the simulations generated by NS. NS was developed using a dual programming language approach: OTcl for flexibility and C++ for efficiency. C++ is used for very frequent and persistent functions, like packet transmission and reception. In the other side, OTcl is used for dynamic protocol object configuration, source traffic pattern specification and topology definition. OTcl defined objects have a high update rate.

The SSFNet[7] is a mature simulation tool available since 1998. Most of components are distributed using open source, but there are some commercial components. This package is built from the following components: the simulation core, written using Java and C++ named SSF(Scalable Simulation Framework), a language to describe the simulated network model, DML(Domain Modelling Language), and an integrated development environment (IDE) with a set of tools for easy development.

GloMoSIM is a scalable simulation environment for wireless networks, with plans to include also wired and hybrid networks support. It is a parallel discrete event simulator. A commercial version named QualNet is available, but research and academic organisations have free access to source code and binaries.

The OPNET[6] is a commercial simulation package developed at MIT. Even without access to the kernel code, needed for some kind of experiments, it provides a friendly interface which enables the design of efficient simulations for large networks. Furthermore, OPNET provides a large set of documentation. When compared with NS-2, OPNET is a more complete package including analytical tools, discrete event, fluid and hybrid based simulations.

Fluid based simulations are an attractive choice to simulate inter-domain routing large scale networks topologies. However as large topologies can be reduced to simpler ones with identical graph properties and our target research focus is the behaviour of the protocols, packets based simulations seems appropriated. So, to benefit from previous experience NS-2 has been selected.

3. MULTICAST ROUTING IN NS2

To run a simulation in NS-2, it is necessary to create a simulator instance, an object of the *Simulator* class. The simulator core is an event scheduler responsible for manage a list of time stamped events which will be executed by order. After that it is necessary to create a network topology: a set of nodes and links between them.

A node is a set of objects of *Classifier* Class, called classifiers, which are responsible for receive and retransmit packets. The entry point of a node is identified by the *entry* variable which points to the first classifier of the node. A data packet generated by a local agent or received from the exterior, is always delivered to this first classifier which must extract the destination address of the packet and do the first classification: determine if it is an unicast or multicast packet. It is a multicast packet the packet is delivered to a multicast classifier which is referenced by a variable called *multiclassifier*. This classifier has the multicast routing table with different entries indexed by (S,G) or (*,G). Associated to each multicast routing entry there is a replicator which has the capability of send a packet to multiple outgoing interfaces.

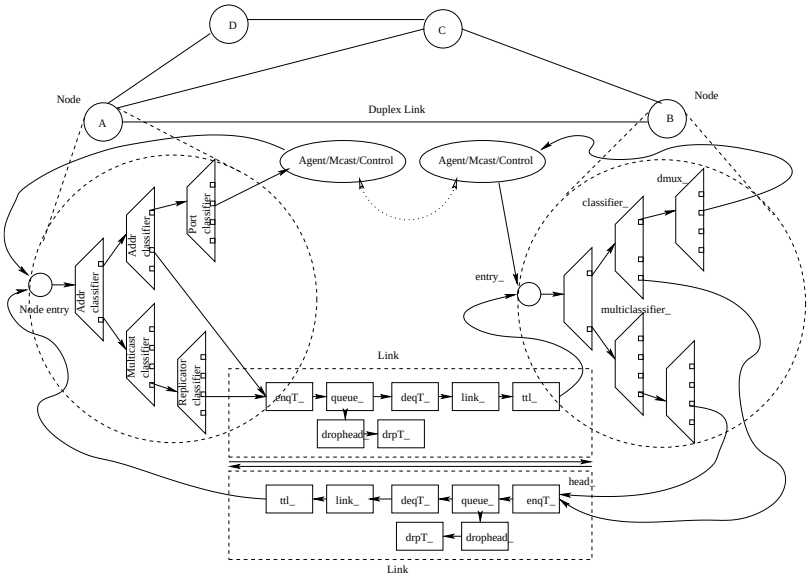
To complete the topology, a set of links between nodes must be created. When creating a link between any pair of nodes, it is possible to specify the type of the link, its bandwidth, end-to-end delay and queue type. A link is a collection of connectors. The first one is referenced by the variable $enqT$ and its main function is to register the packet entrance into the link queue. The connector *queue* is an instance of class *Queue* and its main function is to manage a waiting queue. If a packet is dropped the connector $drpT$ is called. Otherwise, if a packet is scheduled the connector $deqT$ must register the event and the connector *Link* is called in order to take into account the propagation delay. Finally, the last connector is *ttl* and it is responsible for decrementing the TTL (Time-To-Live) packet field.

When dealing with routing, besides classifiers in all nodes, it is necessary to create Routing Agents and to implement a Route Logic Module. Routing Agents are responsible for exchanging routing messages with routing agents in other nodes. With this information the Route Logic Module computes the best routes for all known destinations and updates the routing tables in each classifier. Figure 1 presents a simple topology with 4 nodes and shows the packets path (unicast and multicast) between two of them.

4. QOS AWARE MULTICAST ROUTING PROTOCOLS

Most of inter-domain multicast constrained multicast routing proposals are based on path probing strategy for scalability reasons. This type of strategy is better suitable for large scale networks, because it does not require the maintenance of global state information in the nodes. According to path probing strategy the path searching process is initiated by the new receiver which explores different alternative paths and evaluates them in terms of how well do they fulfil its requirements.

Existing path probing multicast routing protocols differ from each other in the way multicast routing distribution trees are



1: Data Path between nodes in NS2

built. YAM [1] builds shared trees having the capability to provide multiple routes to connect a new node onto an existing tree. In order to find multiple alternative routes from existing tree nodes to the new receiver, YAM relies on flooding. QoSMIC[4] tries to alleviate the YAM flooding behaviour by introducing a new element, the Manager Router. QoSMIC uses two different procedures to find a feasible tree branch: a local search and a multicast tree search. Local search is initiated by the new member router by flooding Bid-Req messages to its neighbours with scope controlled by TTL. Any in-tree router that receives a Bid-Req message becomes a candidate router and replies with a Bid message forwarded to the new member. The Bid request message, on its way, collects information about the path that can be used for selection purposes. The multicast tree search occurs at the same time, initiated by the Manager Router. After receiving a M-JOIN request from a new receiver, Manager Router sends a Bid-Order message to a set of in-tree routers, that became candidate routers and sends Bid messages exactly as described for local search procedure. Figures 2a and 2b illustrate the YAM and QoSMIC tree construction process.

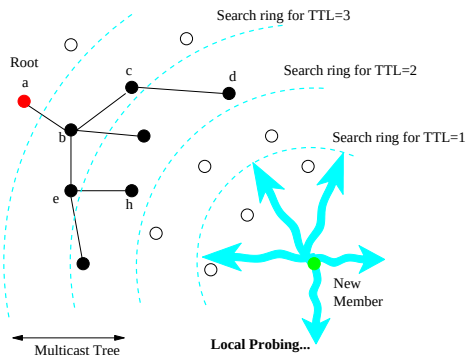
PAQoSIDMR[2] is a path probing multicast routing protocol that takes network asymmetries into account. The right way to deal with network asymmetries is to start the tree construction from its root towards the new leaf member, but this solution causes a greater join latency. In PAQoSIDMR this problem is addressed building multiple directed shared unidirectional multicast distribution tress, one for each multicast group. To avoid tree root routers overload join requests are handled in a distributed manner by the first in-tree router that receives them, thus relieving the tree root of this task. In order to increase the possibilities of finding a feasible path, a controlled number of retries may be conducted by other in-tree routers if the first in-tree node fails to find a feasible tree branch to join the new member.

Figure 2c illustrates a tree branch construction as done by PAQoSIDMR. The new receiver host must send a join-request message specifying its QoS requirements through the tree-root of the multicast distribution tree for that group (the tree root is a well known pre-defined node). The join-request message is forward hop-by-hop until it reaches a router already in the multicast distribution tree. This router will then initiate the new tree branch construction by sending join-answer probing messages back to the new receiver. Before forwarding join-answer messages, all routers must collect dynamic QoS metrics and append them to the message. If the accumulated path QoS metric does not meet the QoS requirements included in the request, the join-answer message must be discarded and a Nack message is sent back. Any join-answer message that reaches the new receiver contains information about a feasible tree branch. A selection procedure must be executed to select one of them according to certain criteria and finally an Ack Message establishes the new tree branch including the necessary state information in each router.

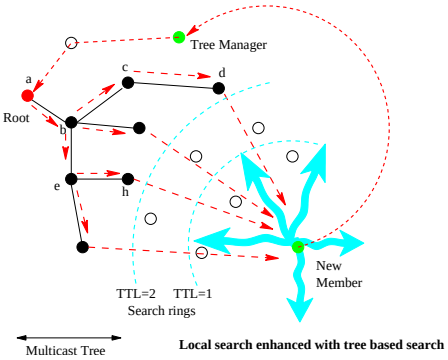
If none of the possible alternative paths can meet the QoS requirements specified, the in-tree node that received the join-request message and led the new branch construction can detect branch setup failures as it receives Ack and Nack messages. If a Nack message arrives for all join-answer messages sent, the router forwards the original join-request message to its upstream-neighbour. In extreme situations with consecutive retry failures many in-tree routers may be involved in the join procedure, which may result in a very large join latency. In order to control better the join procedure, a retry counter is included in the join-request to reduce the number of retries to an acceptable limit.

5. SIMULATION ANALYSIS

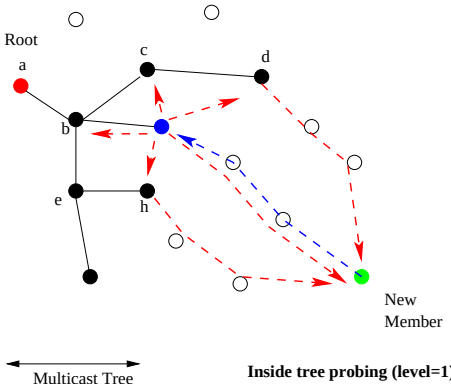
NS-2 simulations are written in Tcl files that can be manually edited or built by scenarios preparation scripts. A simulation Tcl file contains code to build the topology, set link and node parameters an run the simulation. It is basically a set of initial events, tagged by virtual simulation time, that

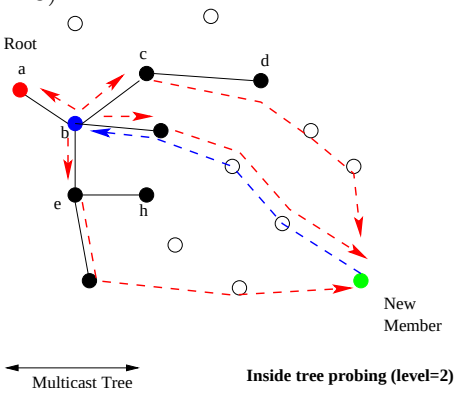


(a) Local probing (YAM)



(b) Local search and Tree based search (QoSMIC)





(c) Inside tree probing (PAQoSIDMR)

2: Path probing strategies schematically compared

are injected in the scheduler event list. When the simulation starts, each event is processed by simulation time order. Simulation ends either because there are no more events in the list or because simulator executes a specific stop event.

Each simulation produces results that usually demand for further processing, in order to make them useful. NS-2 can generate by default two files, containing all packet level events. For each packet generated during simulation, the simulator logs all steps it takes in the simulation topology: enqueue, dequeue, arrival and drop. The two files contain the same type of information but in different formats. The one with the Nam extension can be used to build an animation.

Animation is really useful on initial development state. At this phase it is important to define a small fixed topology with a simple carefully planned scenario. This scenario results can be used in the future to check if code changes are affecting results for the test topology. When a patch code is done, one should perform a simple diff operation between results produced on the test topology and the ones previously stored. This allows the detection of good or bad code patches.

But once the protocol implementation is stable, animation is no longer useful. At this phase one needs to prepare bigger topologies and complex simulation scenarios. Results can be obtained by two different ways: simple analysis of the trace file or by forcing special outputs in code implementation. The best way may be the combination of the two. Arrange for special log lines at important moments: a node joining or leaving a group, a link included or excluded from the multicast tree. Maybe one could also collect and dump more useful data like the QoS metrics observed on links, cumulated values for a tree branch, number of retries in probing phase, etc. Many of this values are easy to obtain because they are already available in protocol internal state variables, but of course they still demand for further post-simulation processing, in order to get average values between multiple simulations. Logs should be produced using `annotate` trace method, to keep them in the global trace file with a known syntax.

With this strategy, one should run a set of 100 or more simulations for each simulation scenario established, process the huge trace files produced for each simulation in order to produce simple text files with average values for most relevant metrics defined (TreeCost, QoS, etc). Graphics can be produced using `gnuplot` or similar tool. We have selected the `R` package for graphics.

5.1 Multicast routing strategies used in simulation analysis

Simulation is, in fact, a tool that one uses for a certain purpose. One simple goal is to achieve an implementation as a simple proof of concept. In this case, it is in fact a protocol specification tool. But it can be used for protocol analy-

sis. In the example case, three QoS aware multicast routing protocols have been compared: PAQoSIDMR, QoSMIC and YAM. A priori we already know that they have characteristics making them comparable. In fact they all use a path probing strategy to achieve QoS. So the simulation must be focused only on the probing method efficiency and cost. Many variables however can affect the results and it is not practical to make them change one at a time between simulations, because of the huge number of combinations. Here we have to identify a small set of variables that have predictable impact on the probing results, like inter-domain topology, link congestion, group membership, and plan a reduced set of experiments. It is however possible that the first set of results points us for new experiments, because some variable not initially considered is affecting results.

So, lets pay attention to the path probing strategies. The oldest approach is to perform local probing using search rings (YAM). We know it works well if groups are dense and the new member is not located far way from the multicast tree already built. It may fail with small TTL values (expanded ring size). Recommended value is 5 but we don't expect it to behave well on inter-domain topologies with sparse groups. We have to test this by changing both topology and group sizes. Group size can be changed during a single simulation by forcing join and leave operations randomly. Topology size can be changed using topology generation tools.

The second routing strategy (QoSMIC) is an enhancement of the first one that introduces a second probing phase, conducted by a tree manager and started by several in-tree nodes. The size of the tree and the position of the tree manager may affect results. This phase is only a complement of the local search when it fails. So they are really not comparable in this way, unless one reduces the time spent in the first phase. Perform local search with TTL=2 and move on to multicast tree search after failure. But we have to introduce some variants to make comparison more reasonable. One could avoid local search and perform only tree probing (QoSMIC-mcast), or, use the best possible result for this strategy: full tree search (QoSMIC-full). Every possible node already in tree launches a probing packet towards new member. We can also try to have the best possible result with YAM by expanding TTL until success (YAM-full).

Having the best possible results for each routing approach, it is also a good idea to have the worst case scenario. In this case, the worst possible approach is to use a multicast routing protocol unaware of QoS, that does not do any path probing, like PIM-SM[5]. While there are many results published using some or all of those routing protocols they are not easy to use without repeating the experiments. To reuse and share results a set of improbable conditions must happen: code access, code portability between different NS-2 versions, topology/scenario and result files available. Even with all that available, metrics used could differ.

Te fourth routing strategy (PAQoSIDMR) pretends to achieve better results in inter-domain level by avoiding all local expensive searches. New member tries to find any in-tree node as fast as possible by sending a control packet towards root. Probing is done by that node, and by other inner in-tree nodes in case of failure, thus avoiding the complexity of having a tree manager. It is expected to behave well even when groups are sparse in inter-domain topologies. Table 1 summarises this.

Protocol	**Comments**
PAQoSIDMR	Retries in case of failure NRetries=2
YAM	Maximum search ring TTL=5
QoSMIC *standard*	Local mode with TTL=2; Tree search mode with 1/3 candidate fraction, directivity and local minimal enhancements
QoSMIC-mcast	Multicast tree search only (with same parameters)
QoSMIC-full	Multicast tree search only with all in-tree nodes as candidates...
PIM-SM	PIM-SM *standard* version

1: Simulation analysis: multicast routing strategies used

5.2 Hierarchical and Network Topologies

Simulation also makes possible to run tests in several different topologies in an easy way. Topologies can be obtained by mimic of real ones or algorithmically constructed from scratch. There are several good topology generators (GT-ITM, INET, BRITE, etc.) and also a strong effort to further improve them. The goal of a topology generator is to build topologies that look like the real ones at least in their major characteristics. This implies that we must first know well the real Internet topologies.

For the analysis of the multicast routing strategies, as well as for other routing strategies in general, topology is a key element. In our example its importance is even bigger because there is a claim that one can build a better solution for the inter-domain scenario. This means that the top level graph of Autonomous Systems (AS) and their relationships can influence the efficiency and efficacy of the routing strategy. The size of the topology is also important because it has a strong impact in inter-member distance in the topology. Can we really perform an expanded ring search on a big topology? Probably we don't need simulation to conclude that, but we sure need it to establish the difference between probing from inside or outside the multicast tree.

At inter-domain level we have to construct topologies, because it is not possible to mimic Internet topology, and there are no guaranties that generated topologies have the characteristics of the real one. Topology generators try to ensure that constructed topologies verify all power laws. Techniques used are therefore different for straight flat topologies and hierarchical transit-stub AS ones. Heuristics improve the results. But perhaps the best solution at the moment is to try to use as many generators and topologies as possible. A practical issue that we need to take care of is the adjustment of the tool outputs to Tcl. Most of the generators output in NS-2 friendly format, but some minor changes can be turned in the source code.

INET generator has been excluded because the smaller inter-domain topology that it constructs must have at least 3017

Name	Generator	Parameters
Flat_3.5	GT-ITM	Method: *geo*; 100 × 100; Links: *Waxman*; $\alpha = 0.033$
Flat_6.4	GT-ITM	Method: *geo*; 100 × 100; Links: *Waxman*; $\alpha = 0.066$
TS600	GT-ITM	Method: *transit-stub*; 3 stub-AS for each transit-AS;
BRITE100	BRITE	Method: *AS + Waxman + Incremental + Preferential*; HS: 100; LS:100; N: 100, $\alpha = 0.15, \beta = 0.2$

2: Simulation analysis: topologies used

nodes. It is out of the question to even load such a topology in NS-2 on a normal workstation with a routing protocol activated on each node. For multicast simulations this means at least two protocols and two routing tables per node. Simulator has to create several objects per node, and packet level simulation becomes real painful. And that is when everybody starts looking for other approaches for simulation, such as the ones based on fluid dynamics. However, when QoS is the goal, like in this multicast routing experiment, localised congestion, policy limitations and others are very important issues to consider. Packet level simulation ensures that they are not excluded.

While the focus is on inter-domain one can not exclude from initial analysis the behaviour on a standard flat topology. Table 2 gives the complete range of topologies considered in our experiments.

5.3 Traffic Models

Besides topologies, that model the static objects of a network, we must also bring traffic to the simulation in order to reproduce the dynamic behaviour of a real network. Once more we would like to have traffic patterns similar to the real ones. One common way to do it is to use traffic generators like the ones included in NS-2. A traffic generator generates packets with a given size and puts them in the topology with a certain rate. For multicast experiments we have used the CBR traffic generator configured to produce a 210 byte packet at each 420 ms, thus giving a small constant bit rate of 4 kbps. Traffic generation greatly degrades the simulation performance, because the number of resources needed is directly proportional to the number of packets that traverses the topology. Each packet is a very small object that does not really moves in the topology. Travelling is simulated by moving a pointer to the packet from node to node. But that does not solve the problem.

Another way to deal with the traffic is to reproduce only its effects. Packets traversing a link shorten its available bandwidth. Congestion introduces delays and losses that can be registered only. On can build a traffic load generator that reflects the load effects on all objects. This can be done once only, at the beginning of a simulation, or periodically through the simulation. Some topology generators, like BRITE for instance, can use a distribution function (normal, exponential, heavy tail) to generate the available bandwidth on each link. This methodology can be followed and further improved.

Our strategy in here was to generate multicast traffic in each group, with CBR, and to avoid any other traffic as much as possible, in order to be able to deal with bigger topologies. Available bandwidth on links was artificially changed at the beginning of each simulation and varies from simulation to simulation. Multicast packets also help tracing tree shape in NS-2 trace files.

5.4 Simulation Scenarios

Since our goal in the simulation is to compare the four different multicast routing strategies we must deal with group dynamics in the simulation scenario. Group dynamics are defined by three major parameters: the number of multicast groups in each instant, the number of member that join each group and finally the sequence of join leave operations. Too many things. We can pay attention only to one group at each simulation with no problem, considering that other do exist but are considered as other traffic. Group size can be varied from small to very large in the same simulation by increasing or decreasing the number of join operations. In this way we have reduced the problem to a sequence of join leave events and the node that must perform it.

Simulation scenario is defined in the simulation script written in Tcl and there is no problem to automate it according to this (or other) scenario. While this can be reviewed at any time it is very important to clearly define it before simulation in order to reduce the number of available distinct results and also to make things easier in result interpretation. Group dynamics has a strong effect on multicast trees constructed and good result interpretation demands for a good knowledge of it.

In our simulations we rooted each tree on one randomly chosen node. Nodes randomly chosen join the group at regular time intervals until having 60% of all nodes as receivers. In this set of experiments no leave operations were generated since the focus of the simulation is the path probing strategy. Path probing occurs at join time and depends on the tree already constructed and the position of the new member. Leave operations would only impact the tree size which is already changing with the join operations.

Another important issue in this specific simulation scenario is the QoS needed and really obtained by each member. Again a lot of things to deal with. If a member expresses QoS needs that routing protocols can not achieve, a decision needs to be taken: whether or not to connect the new member to the tree. One possible approach is to understand the requirements as mandatory and measure the connection success rate. To reduce the number of variables changed per simulation instead of generating QoS requirements per member, we decided to always join each member to the best available path. Different strategies can still be compared with each other by success rate, but success is measured as the percentage of nodes that achieved a QoS greater or equal to each possible value.

5.5 Metrics

After having defined what to simulate and how, the next step is to define what to measure and how. A small set of metrics should provide a clear view on the efficiency and efficacy of each multicast routing strategy.

Some metrics are more specific and related to QoS awareness while others are more general and highlight tree characteristics. One metric already lined up in the previous section is the QoS obtained by each member. Each routing strategy, after path probing, provides the new member with the best path available. In this set of results the best possible path is the one with the higher available bandwidth, and QoS obtained is expressed as the available bandwidth on the entire branch from tree root to the new member. Success measures can be stressed graphically with the percentage of members versus the QoS obtained.

Metric	Meaning
ObtainedQoS	QoS value new member gets when joining tree

(a) Efficacy: ObtainedQoS

Metric	Meaning
NumLinks	Number of links in multicast tree
SumLinks	Sum of link cost for all tree links
ReplCost	Average packet replication cost
TreeHeight	Average links per tree branch
TreeCost	Average tree branch cost

(b) Efficiency: Global Tree Cost

Metric	Meaning
Msgs	Average number of control msgs per request
PathsProbed	Number of paths probed per request
Latency	Join probing time (best answer only)

(c) Efficiency: Control Overhead

3: Simulation analysis: metrics

Efficiency measures are divided in two categories (table 3): tree cost and control overhead. Global tree cost can be expressed in a simple way as the number of links that belong to the multicast tree. This number can be obtained by explicitly counting them in code implementation whenever a link is added to the tree. While easy to obtain its expressive power is reduced because we don't know how tall or fat is the tree or the real link cost. Another measure, also global, can be obtained by adding all tree link costs. It suffers however from the same limitations of the previous one related to the tree shape, but takes real link cost and not simple hop count measures.

But in this category the best metric is the replication cost because of its improved semantic. Instead of counting the number of links or its cost, one can use the NS-2 trace file and follow one packet from source to destination. Each time the packet is replicated at some node, one most also follow each replica, cumulating the cost of the link followed. At first it may look exactly like the sum of links but it is not because it counts only branches really used by data. The average number of links per tree branch and the average tree branch cost give a perspective on the tree height.

The second category of efficiency measures accounts for the multicast tree construction effort in terms of the number of control messages generated by each operation. Some approaches achieve better efficacy but at the expense of a greater construction effort, so to be fair on result analysis all metrics are needed. Since the focus in result analysis is the path probing strategy one may complement the analysis with two specific measures: number of paths probed and time spent in the entire probing phase. Again, these two metrics (PathsProbed and Latency) can be collected and dumped in the protocol implementation while the number of control messages is better obtainable by counting them in the NS-2 trace file.

6. RESULTS

Figure 3 presents a set of results obtained. All results presented are an average of 100 distinct simulation runs for each topology and scenario.

The upper two graphics (3a and 3b) show efficacy measures on flat and hierarchical topologies. The first one shows the QoS obtained for topologies with 100 nodes generated by the GT-ITM tool while the second presents results also for 100-node hierarchical AS topologies generated by BRITE. Available bandwidth varies in both cases between 0 and 10 Mbps. PIM-SM is, in both cases and as expected, the bottom line since it does not use any path probing. By looking at the graph we know that 50% of the nodes that joined the group achieved only less than 3 Mbps, either in flat or hierarchical topologies. The relative positions of the other protocols remain the same in both graphics showing no strong evidence that the type of topology affects results. This reading can be later transformed into a conclusion if confirmed by further experiments. Figure 3b includes more lines than 3a because we are trying to see how PAQoSIDMR parameter number of retries influences the results.

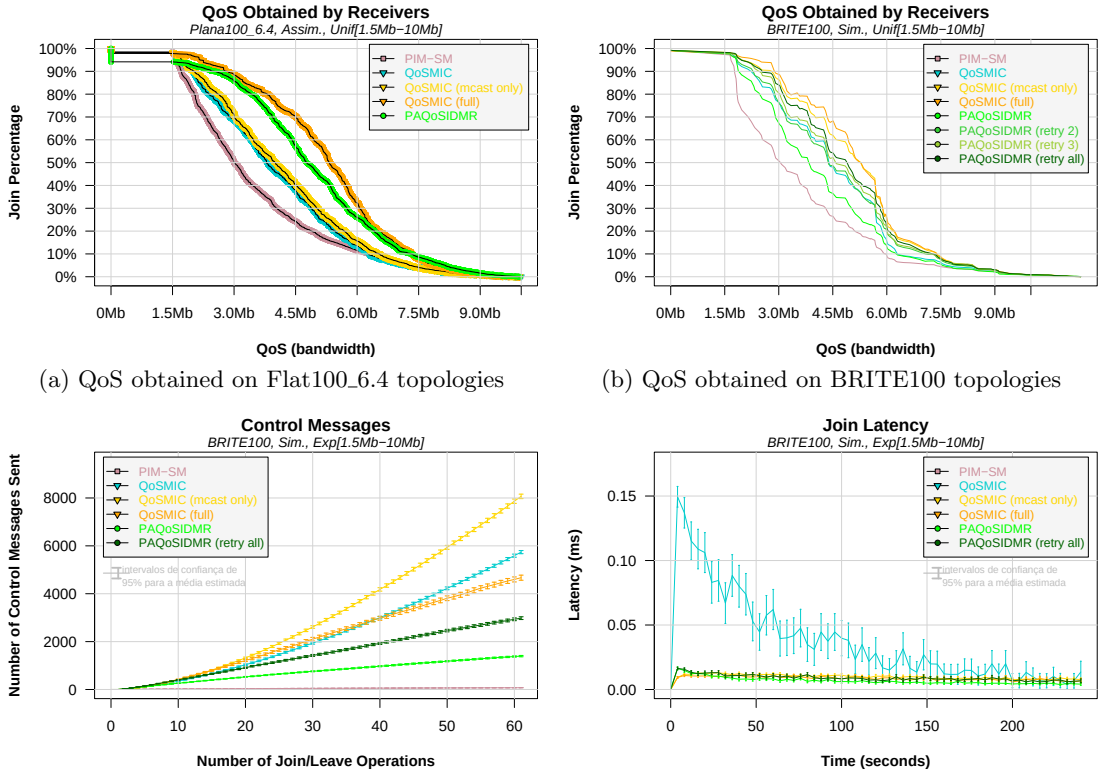
The two bottom graphics (3c and 3d) show the efficiency metrics. The first one is the average number of control messages per join operation and the second one shows the average join latency. They confirm that local search can be very expensive specially when the tree is small and new members are far most of the time very far way from it.

7. CONCLUSIONS

This paper reports the experience gained by the usage of simulation tools in multicast routing with QoS. It is therefore not focused on the real simulation results reported but on the process used to obtain them. NS-2 has been used for several years and by thousands of investigators with success and the new NS-3 version is already in beta stage.

One known difficulty founded is that code contributions usually stay sticked to a NS-2 version. The code that implements the four mentioned multicast routing strategies runs well on NS-2 2.1b8 but we haven't tried to patch it for the latest 2.30 version. That happens with almost all contributions when they start to receive less attention from the community.

Many QoS related functions should be integrated in the NS-2 core to avoid custom adjustments. That already happened with some important pieces of the DiffServ quality of service model, like RED queues, Edge/Core nodes, etc. QoS monitoring on links is easy to achieve but users always include



(a) QoS obtained on Flat100_6.4 topologies

(b) QoS obtained on BRITE100 topologies

(c) Control messages overhead on BRITE100 topologies

(d) Join Latency on BRITE100 topologies

3: Simulation results: an illustrative snapshot

private variables and procedures to do it, making it more difficult to share and to upgrade. The same should be said about inter-domain issues. The definition of AS and a BGP-like implementation are missing and should be provided as part of the major distribution to make the usage of NS-2 for inter-domain routing more easy.

Another important issue, this one not related to NS-2, is the difficulty to share and compare results between users. One can observe interesting results published on an article but the effort to reproduce is huge. With papers of six to ten pages long, it is not possible to provide all details and the authors must make their point and prove it. Details are sometimes forgotten. The learning curve of a powerful tool like NS-2 is high but the results curve is much higher. After simulation, and also not related to NS-2, comes the need for post-processing tools. The natural tendency to build post-processing scripts à la carte or to try to use some general purpose analysis tools distracts users from their real goals. In this point it is up to each users community to share experiments, to enumerate the needs and work on the tools that can satisfy them. On the evaluation of QoS aware multicast routing strategies, packet level simulation as the one provide by NS-2 plays is an indispensable tool.

8. REFERENCES

[1] K. Carlberg and J. Crowcroft. Building shared trees using a one-to-many joining mechanism. *ACM Computer Communication Review*, pages 5–11, 1997.

[2] A. Costa, M. Nicolau, A. Santos, and V. Freitas. A new path probing strategy for inter-domain multicast routing. *Next Generation Internet Networks, 2005*, pages 9–15, 18-20 April 2005.

[3] K. Fall and K. Varadhan. *The NS Manual*, Jan. 2001. URL=http://www.isi.edu/nsnam/ns/ns-documentation.html.

[4] M. Faloutsos, A. Banerjea, and R. Pankaj. Qosmic: Quality of service sensitive multicast internet protocol. In *SIGCOMM*, pages 144–153, 1998.

[5] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas. Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). RFC 4601 (Proposed Standard), Aug. 2006.

[6] O. Inc. Opnet - network modeling and simulation products, 2005. URL=http://www.opnet.com/.

[7] A. Ogielski, D. Nicol, and J. C. have. Ssfnet - scalable simulation framework, 2002. URL=http://www.ssfnet.org/.

[8] V. Paxson and S. Floyd. Why we don't know how to simulate the internet. In *WSC '97: Proceedings of 1997 Winter Simulation Conference*, pages 1037–1044, Atlanta, GA, USA, Dec. 1997. ACM Press.

[9] A. E. Rizzoli. A collection of modelling and simulation resources on the internet, 2005. URL=http://www.idsia.ch/ andrea/simtools.html.