Quality of Service Aspect for BRAIN Architecture

Cédric Adjih[1], Khaldoun Al Agha[1], François Dumontet[1], Philippe Jacquet[1], Alberto López[2] and Laurent Viennot[1]

[1] INRIA - HIPERCOM, B.P. 105 78153 Rocquencourt cedex France
Phone: 331 39635905, Fax: 331 39635566, E-mail: {first name. last name}@inria.fr
[2] Universidad Politécnica de Madrid, Departamento de Ingeniería de Sistemas Telemáticos, ETSI Telecomunicación · Ciudad Universitaria · E-28040 Madrid
Phone: 034 913367366 + 442, Fax: 034 913367333 E-mail: alberto@dit.upm.es

ABSTRACT

In this paper, we present different aspects of Quality of Service that should be adapted to the BRAIN architecture. Several parameters and policies of QoS are depicted. Also, the paper shows the dynamic adaptation of these parameters in the context of BRAIN.

I. INTRODUCTION

QoS refers to the capability to provide a better level of service to selected network traffic. This level of 'service' is usually defined in terms of some basic performance criteria or parameters such as bandwidth allocation, loss ratio, delay or jitter. So, QoS is provided when there exists a level of assurance that service parameter requirements can be satisfied.

A number of QoS protocols have evolved to satisfy a variety of applications QoS needs in opposite to fixed IP 'best effort' networks. On the other hand recent progress in computing technology and wireless digital communication has made portable computers easily available. Mobility of hosts has a significant impact on the quality of service provided to a real-time application, turning those protocols useless in a mobility environment.

One of the main challenges of BRAIN is to solve the problem of the QoS management in the context of mobility. When a mobile roams between two base stations the data flow may often change. As a result, the propagation delay of packets may change. If the new base station has different utilization, the available bandwidth may not be sufficient to provide the throughput it was receiving at the previous location or even connection can be disrupted during handover. Host may have to adapt its QoS with the available resources in the new base station. QoS re-negotiation is therefore the key for solving this problem. The BRAIN architecture has to support three main features:

- Inter-domain mobility management: By inter-domain mobility we understand the mobility where the host registers into a new IP domain and changes its local IP-address. By extension we call an inter-domain mobility the case where the host switches from one IP access (for example UMTS) to another IP access (for example Hiperlan). In inter-domain mobility the continuity of service is less crucial since the server and/or the intermediate routers would need to reconfigure their client addresses and the host would probably be out of reach while in motion. Inter-domain mobility would be managed via Mobile IP.
- Intra-domain mobility management: By intra-domain mobility we understand the mobility within a same domain where the host does not need to change its local address when moving between base stations. The continuity of service must be maintained during handovers. The management of mobility with continuity is one key challenge since actual IP implementation does not allow real-time mobile tracking.
- QoS management: The broadband nature of the wireless interface provides great opportunities for the development of wireless multimedia applications with a sensible need of QoS management. Furthermore the network is also open to mobile networks and calls for a differencied access control between for example premium and standard clients. The QoS management beyond base stations can be handled by *diffserv* policy. One key challenge is management within a base station and more crucially the QoS management during handovers. For example a multimedia server may have to significantly reduce its bit rate when the mobile switches from a base station with low activity to a base station with high activity and less available bandwidth.

A. 3D BRAIN scenarios (QoS, mobility, priority-cost)

The scenario is characterized by three sets of parameters:

- the application QoS parameters (bandwidth, delay, renegotiability)
- the user mobility within the network (e.g. fixed, nomadic, walking)
- the user cost and privileges (high priority user, gold, silver, best effort)

A priori these three sets of parameters are set independently: there is no relationship between QoS required from application (VoD, gaming) and the user mobility (fixed, walking, driving), and the user privilege for example a doctor may walk close to a patient lying in a mobile bed and check brain surgeon video records (high QoS, high mobility, high privilege). A patient may want to watch a VoD (high QoS, low mobility, and low

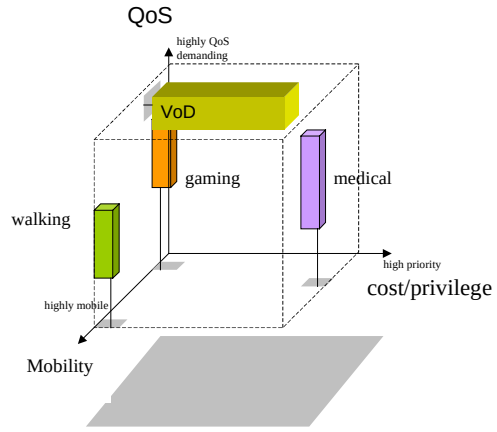
privilege). A rich patient may want to see a VoD with higher privilege, etc.

Indeed, none of the three sets could be reduce to a single scalar value, it is very convenient to identify the various scenarios as elementary volumes in a 3D cubicle display. Therefore we arbitrary set an origin and three axes:

- one for mobility (from fixed to high mobility)
- one for QoS parameters (from low demanding QoS to high demanding QoS)
- one for costs/privilege (from no privilege to high priority)

With respect this very simplified description one can display the three main scenarios types:

- gaming: medium-high QoS, low privilege, no mobility
- medical: medium-high QoS, high priority, medium mobility
- walking: low-medium QoS, low privilege, high mobility
- VoD: high QoS, low-high privilege, no mobility



B. QoS management in BRAIN

The QoS parameters are the parameters that the user is able to quantify about the network in order to obtain the success of its distributed application.

The QoS parameters are used to establish the contract between the user and the network for the completion of a service. A single set of QoS parameters is used to define the properties of a flow, which is the most convenient level of network management. However an application may need several flows (i.e. video with audio), therefore an application may need several sets of QoS parameters.

We distinguish two levels of QoS parameters. First level parameters address the QoS of a flow between two negotiation phases (basic QoS parameters); the second level addresses the negotiation phases (negotiation QoS parameters). The negotiation QoS parameters contains a list of basic QoS parameters (QoS levels) with some informations about timing, mobility awareness and application reactivity to change.

A certain number of LANs (based onto multiplexing techniques) allow a real time control of the maximal bandwidth that every client connected can use. Thus, a centralized supervisor called QoS broker will manage and allocate bandwidth to each connection. Each user in need of QoS will make reservation to this QoS broker. The user request will contain the QoS parameters described below. This configuration is an efficient way to provide QoS into a LAN. Principles leading us are, setting the QoS mechanisms transparent for the application.

II. BASIC QOS

A. Basic QoS parameters

User/application QoS parameter: the basic QoS parameters are typically:

- Bandwidth
- Service duration
- Maximum Delay
- Maximum Jitter
- Maximum Loss rate
- Type of service (guaranteed, predictable...)

1. Bandwidth parameters

The bandwidth is classically defined by three parameters: the *peak* rate, the *medium/sustainable* bit rate, and the *variance* rate. Recently, proposals have been done to replace the medium and variance bit rate parameters by a single parameter: the *equivalent bandwidth*, which is more appropriate with leaky bucket strategy [7]. The equivalent bandwidth is defined with respect to the overflow probability in a leaky bucket. If the traffic were Poisson with constant rate λ, then the probability of overflow $P(B,W)$ in a leaky bucket of size B and service bandwidth W is asymptotically equal to $\exp(-\frac{W}{\lambda}B)$ [7]. The equivalent bandwidth λ_e, when it exists, is the quantity $\lambda_e = W \lim_{B\to\infty} \frac{B}{-\log P(B,W)}$ [7].

The equivalent bandwidth is very useful in order to fix the leaky bucket parameters, namely the total bucket size B and the total service rate W_c for this class. In general λ_e is larger than the medium rate. One possible management is to make $W_c = (1+\varepsilon)\sum \lambda_e$ [7].

The peak rate is needed in order to fix the maximum value of service rate of this class:

$W_c = \max\{\text{Peak rates}\}$.

The peak rate is not necessarily equal to the maximum instantaneous bit rate of the service. For example in Video-on-Demand (VoD) services, the client stores incoming video packet in a buffer whose aim is to absorb packet delivery jitter. Packets are de-stored after a delay T, which corresponds to the maximum absorbable jitter. In this case the peak rate is computed according to the following formula:

$\max_x\{\frac{1}{T}\int_x^{x+T}\lambda(t)dt\}$ where $\lambda(t)$ is the expected instantaneous bit rate of the server at time t [7].

2. Service duration

An expected service duration could be specified by the user. This can be done in two different ways: a 'fixed' one as in switched networks where the time defined by explicitly sending a reservation request and later a reservation finish messages. Another way is a 'soft state' one as in RSVP IntServ basis, where user specifies a refresh period (10 secs, for example). In this case, a service which would actually last more than the expected duration allowed duration will need a refresh declaration, to be initiated by the client/application or by the local QoS broker. In other case, reservation is finished implicitly.

3. Maximum delay

For interactivity purpose, a short delivery delay may be requested. Telephone needs a delay delivery of the order of few 10 or 100 msec (depending on the use of echo cancellation or not). This indication will be provided in the maximum delay parameter. This will help the QoS broker to fix the class of service (expedited services or other) and to fix the bucket buffer size.

4. Maximum jitter

Video on demand, audio on demand need that jitter in packet delivery be controlled. The maximum jitter will be determined by the client and the application according to the size of their delivery buffers. The QoS broker will use the maximum jitter parameter in order fix the service class and the bucket size and bucket server rate.

5. Maximum loss rate

The loss rates (in packets) will indicate to the QoS broker the maximum tolerable packet loss. For example telephone can accept an uncoverable loss rate of order 10 %. However, the service could support *recoverable* packet loss. In this case the service could accept packet loss provided and a higher level data recovery protocol will retransmit packets (like in RealServer streaming process). If the client indicate recoverable loss rate, the QoS broker will need to expand a little more the bandwidth requirement. For example a service that need a 1 Mbps constant bit rate with 10 % recoverable packet loss would imply for a 1.1 Mbps bandwidth reservation, if packet loss occurs. In any case (uncoverable and recoverable) the packet loss should not exceeds the requirement.

B. Policy

Traditional practical admission control algorithms have often been greedy, that is, they often accept a new connection as soon as the scheduling constraints for all the current flows plus the new one are fulfilled. Unfortunately, during time of congestion (such as in occurrences of *Slashdot Effects* [1]), this will strongly favor requests with less bandwidth requirements, along with requests using less links [2]. On one hand, this might be the correct policy, in a similar way TCP congestion control tends to realize *proportional fairness*, which is optimal in some (realistic) sense [3]. On the other hand, different policies can be acceptable; they basically depend on some axiomatic definition of *fairness*, which leave room for arbitrary choice. For instance [4] uses cooperative game theory and shows how different known axiomatic are really part of a specific continuum of policies parameterized by one value. This choice among different fairness should be left, since implementation should not decide on unresolved politics issues.

Moreover, the economical value of one call is certainly not only reflected by its QoS requirements alone: for instance the same call (or piece of information for that matter), can be a life saver for one person, while mostly redundant for another.

Thus, there is a clear necessity to provide some kind of policy enforcement at the admission control level. One promising avenue is, for instance, the simple stochastic control of [5], which allows for simple differentiation between calls, and which will be used. Its details are enumerated in what follows, and later we will describe how it will be extended.

The admission control algorithm goal is to offer statistical guarantees for call rejection rate of a class of the requests. For instance some class of users (paying extra fees), some class of applications (telephony), or some class of calls (important ones) could be favored. The idea is basically to reduce the load from other requests by rejecting an adequate proportion of them *a priori*. Having to compete with less requests, favored requests can get the desired rejection rate. The algorithm relies on four properties:

A way to identify favored connections: favored ones are called *premium*, while the others are called *medium*.

A *parameterized* differentiated admission control enforcing a discriminating policy. Here, it is the application of a simple probabilistic filter before normal greedy admission control. The filter uses a probability *p* (the parameter) in the following way: all the medium requests are rejected a priori with a probability 1-p regardless of the state of the network. The requests surviving this step are then admitted whenever possible.

A way to evaluate the performance of the algorithm, and use feedback from performance mismatch to tune the parameter. The performance of the algorithm is judged from the rejections or admissions in the final greedy admission step: typically rejection indicates a that the load might be too high, while an admission is a signal for existence of room for increasing the load. In this spirit, the parameter *p* is decreased at each rejection and increased at each admission.

A way to configure the tuning step, so that overall performance meets the desired policy. Here the goal is that, throughout the evolution of *p* following the rules of

the stochastic control, premium rejection rate is less or around the defined target rate.

Countless variations of the stochastic control algorithm could be designed using the same principles: classification, parameterized differentiated admission, parameter adjustment.

For *elastic continuous* applications, instead of a pure rejection/admission scheme, it is possible to allocate only part of the maximal reservation the application requirement. Generalization of the described algorithm is straightforward considering that instead of accepting a request for bandwidth *B* with a probability *p*, we now try allocate a bandwidth *p.B* for medium requests, and make a control on *p.B* or on *p*.

The target for premium calls could be any criterion: call rejection rate would transpose to average proportion of accepted bandwidth. For instance one could require that on average 95% of maximum bandwidth requirement of premium requests is allocated to them.

The control on *p.B* would tend to max-min fairness for medium users, i.e., it would tend to allocate equal bandwidth to all medium users. Different bandwidth allocation policies are possible [6]; going further, the notion of utility for a given bandwidth need to be introduced (see next section). A control could then be done on utility, or any combinaison of the utility and bandwidth. Then it is possible to implement some kind of fairness according to game theory and economical principles, such as the ones given in the already cited [4], or others.

III. BRAIN SPECIFIC QOS

This section covers in details what specific parameters can be chosen for the BRAIN architecture:

- QoS levels
- Utility function: each QoS level is valued between 0 and 100, the value is called the utility function
- QoS change reactivity Mobility awareness: motion speed, or the maximum number of expected handovers during service

We call QoS tuple a set of QoS parameters that defines the QoS of a service between renegotiations.

1. QoS levels

The QoS levels can take two forms:

a) a list of basic QoS tuple for the discrete case

b) an interval of QoS levels for the continuous case. Each QoS tuple contains the basic QoS parameters such as bandwidth, duration, delay, jitter, and loss. In the continuous case the interval of QoS levels will be indicated by two QoS tuples:

Desired QoS level	Minimal QoS level
desired bandwidth	minimal bandwidth
desired duration's	minimal duration's
desired delay	maximal delay
desired jitter	maximal jitter
desired loss	maximal loss

Clearly it is very likely that respectively minimal delay and minimal loss actually exceeds desired delay and desired loss, thus there is an abuse of notations. Very likely desired duration and minimal duration will be equal.

By convention one assumes that for all $\mu \in [0,1]$ the QoS broker accepts the QoS tuple made of the following parameters:

- bandwidth = minimal bandwidth + μ *(desired bandwidth-minimal bandwidth)
- duration = minimal duration + μ *(desired duration-minimal duration)
- delay = minimal delay + μ *(desired delay-minimal delay)
- jitter = minimal jitter + μ *(desired jitter-minimal jitter)
- loss = minimal loss + μ *(desired loss-minimal loss)

2. Utility function

In order to help the QoS broker to find the best QoS level per user in a renegotiation phase, it is particularly important to indicate some kind of preference order. One typical way to do so is to define a utility function on the QoS levels. an utility function of 100 on a QoS tuple will indicate that the level is the most desired. A utility function of zero or few units will indicate a less preferred QoS level. A utility function should be attached to each QoS tuple defining the QoS levels.

The first aim of the utility function is to indicate in the negotiation and renegotiation phases the preference of the user in some QoS tuples on other. This may impact the management policy of the network and the QoS attribution policy.

In the discrete case the QoS levels would look like (example of VoD service)

peak rate (kbps)	equivalent rate (kbps)	duration (sec)	delay (msec)	jitter (msec)	loss (in %)	utility
4,000	1,000	10	1,000	1,000	2% (R)	100
1,000	100	10	5,000	5,000	5% (R)	50
100	40	10	5,000	5,000	10% (R)	10

In the continuous case one could a priori assume that the utility value of the QoS-tuple equal to

(1- μ)*desired QoS + μ *minimal QoS is exactly equal to 100μ, but this would provide a lack of flexibility regarding continuous adaptable service. One possible way to escape this dilemma is to define the utility value as a function of μ. To this end a set of standardized utility functions $f(\mu)$ which be defined and the user will have to find the most appropriate one with regard to its application (for example $f(\mu)=\mu$ or $f(\mu)=\mu^2$).

3. QoS change reactivity

This parameter gives the time needed for the QoS broker to achieve the change of its QoS parameters. For example, an MPEG video server would need to wait for

the end of its current group of frame before switching to a new bit rate. A server with a longer QoS reactivity may need anticipated handover when its client roams to a busy cell.

The QoS change reactivity parameter should contain an indication about the way of making QoS changes. There are basically two ways:

- the explicit change (E)
- the implicit change (I)

In the explicit change, the QoS broker informs the server about its new QoS tuple. After the QoS change time, the server should deliver packets according to the new bit rate.

In the implicit change, the QoS broker cannot inform the server about the new QoS levels. The change must be done by forcing the server to adapt itself on the available bandwidth (like in TCP-IP). In particular the QoS broker will force the router to let packet of this flow to get through with the new bit rate. This can be done by killing packet for example, when the new throughput is smaller. Fractal compression streaming of RealServer is acting like that.

4. Mobility awareness

The mobile node can be aware of its mobility and ask the QoS broker to be ready to execute several handover during the service duration. The minimal way of expressing mobility is to provide an indication

- Mobile (M)
- Fixed (F)

In case of mobility, a motion speed indication would be welcome. Providing this motion speed, the QoS broker would be able to identify the potential base station attainable by the user during the service and therefore to make anticipated reservation on these base stations. Of course the additional reservation will impact the billing of the service by the network operator (if any). The motion speed indication could be enhanced by the indication of the motion vector. The use of such QoS parameter will imply that the QoS broker contains a map of the base station coverage in its area. The indication of mobility awareness (M) without speed vector will implicitly contains default values.

Another more complex mobility approach could be similar to the one showed on [8]. For some kind of class service such as Mobility Independent (either Guaranteed or Predictive) which is not supposed to be affected by mobility of hosts, a more predictable mobility of a user is required. It must be able to provide a precisely *mobility specification*, which is the set of locations the mobile host is expected to visit during the lifetime of reservation.

In fixed networks the required resources are reserved at the networks elements along the data flow path. In a mobility environment this data path changes as user moves from one location to another. To provide real-time services to a mobile hosts, specially a guaranteed service, mobility specification is needed to resource reservation along all possible data paths which may be used during connection.

So we can extend mobility awareness parameters as follows:

mobility awareness	Class of service	Mobility specification
M	Mobility independent	<location>,<location>, ...
M	Mobility dependent	N/A

Class of service is extended with bandwidth and QoS change reactivity to match services as guaranteed, predictive and so on. Of course the QoS in a mobility independent reservation is maintained as long as the hosts moves are limited to its mobility specification and it is conforming to its traffic characterization.

In the case of mobility dependent class the host may receive predictive services with high probability but service therefore can not be assured, so it can experience degradation of its QoS even its flow can be dropped.

IV. CONCLUSION

In all networks, the QoS is a crucial problem that should be managed prudently in order to offer perfectly the requested demand. The major aim of this paper was to define QoS parameters and policies that can be applied to the BRAIN architecture. Then, the adaptation of QoS general parameters to a BRAIN context was illustrated.

V. REFERENCES

[1] Stephen Adler, The Slashdot Effect, An Analysis of Three Internet Publications, and Addendum to The Slashdot Effect, Linux Gazette, March 1999, Issue#38 (http://www.linuxgazette.com/issue38/adler1.html and http://www.linuxgazette.com/issue38/adler2.html).

[2] Breslau, L., Jamin, S. and Shenker, S., *Comments on the Performance of Measurement-Based Admission Control Algorithms*, Proc. of IEEE Infocom 2000, March 26-30, 2000.

[3] F. Kelly, Charging and rate control for elastic traffic, European Transactions on Telecommunications, volume 8 (1997) pages 33-37.

[4] Zbigniew Dziong, Lorne G. Mason, *Fair-Efficient Call Admission Control Policies For Broadband Networks -- A Game Theoretic Framework*, IEEE/ACM Transactions on Networking, Vol. 4 No 1, February 1996.

[5] Cedric Adjih, Philippe Jacquet, Philippe Robert, *Differentiated admission control in large networks*, Proceeding of IEEE Infocom 2000.

[6] L. Massoulié, J. Roberts, *Bandwidth sharing: objectives and algorithms*, Proceeding of IEEE Infocom 1999.

[7] F. Kelly, *Notes on effective bandwidths*, in: Stochastic Networks: Theory and Applications (Editors F.P.Kelly, S.Zachary and I.B. Ziedins), Oxford University Press, 1996. 141-168.

[8] Talukdar, A. K., Badrinath, B. R., Acharya, A., "MRSVP: A reservation Protocol for an Integrated Services Packet Network with Mobile Hosts".