A new approach for the prediction of end-to-end performance of multimedia streams

Gerardo Rubino Irisa - INRIA / Rennes
Campus universitaire de Beaulieu
35042 Rennes, France
rubino@irisa.fr

Martín Varela Irisa - INRIA / Rennes
Campus universitaire de Beaulieu
35042 Rennes, France
mvarela@irisa.fr

Abstract

This paper proposes a new and accurate way of predicting the end-to-end performance of a multimedia stream. It consists of coupling an approach we have previously developed, which is able to capture very precisely the way humans assess an arriving stream, with classical performance evaluation models. The former approach can automatically quantify the quality of the connection as humans would do (using statistical learning tools), considering this quantitative measure of quality as a function of measurable parameters of the network and the source; the latter allows mapping this "ultimate" quality value to input parameters such as offered traffic or loads, through, for instance, queuing models. In the paper, we also propose a simplification of the global approach dealing with simple closed formulas. We illustrate our approach with a simple case study of unidirectional VoIP performance.

1. Introduction

In recent years, the growth of the Internet has spawned a whole new generation of networked applications, such as VoIP, videoconferencing, video on demand, music streaming, etc. which have very specific, and stringent, requirements in terms of network QoS. The current Internet infrastructure was not designed with these kinds of applications in mind, so multimedia applications' quality is very dependent on the capacity, load and topology of the networks involved, as QoS provisioning mechanisms are not widely deployed. Therefore, it becomes necessary to take these new applications into account when designing a new network.

The problem is that it is not easy to accurately assess the performance of multimedia applications, as the quality perceived by the end user is a very subjective concept. There are currently three ways to perform this assessment. The first one is to put human observers in front of the streams and to ask them about their perception of quality; this defines what the ultimate quality is, and it is called *subjective testing*. The associated procedures are today well understood; many of them have even been standardized (see next section). The second one is to implement some "distance metric" between the original sample and the one received; this is called *objective testing*. It has been designed to analyze the coding effect on the sequences (not the network impact on them). Unfortunately, when the sequences have been transmitted over a network, the results given by objective measures correlate poorly with the subjective ones (see next section for some references).

The third approach is the one published in [19, 22]. The idea is to train an appropriate tool (a Random Neural Network, or RNN) to behave like a "typical" human evaluating the streams. This is not done by using biological models of perception organs but by identifying an appropriate set of input variables related to the source and to the network, which affect the quality. One of the main characteristics of this approach is that the result is extremely accurate (somehow by construction, as it matches very well the result obtained by asking a team of humans to evaluate the streams). In [22], we applied this method to analyze the behavior of audio communications on IP networks, with very good results after comparison with real human evaluations. In the next section we provide a detailed description of this approach, but let us point out here a useful way of looking at it. The method consists of a procedure allowing to build a function $Q = f(x_1, \cdots, x_P)$ where $x_1, \cdots, x_P$ are source and network parameters (such as the codec, source bit rate, loss rate, jitter, ...) and Q is a measure of quality (such as a Mean Opinion Score, or MOS). The results presented in [19, 22] show that if the stream encounters source and network conditions such that the chosen parameters have values $x_1, \cdots, x_P$, then $f(x_1, \cdots, x_P)$ will be very close to the quality value an average human would give to it.

The main contribution of this paper is the coupling of this technique with standard performance models in order to predict the behavior of the systems, typically for dimensioning or for capacity planning purposes. Let us explain this more in detail. Assume you are in charge of evaluating the performance of some IP network to be deployed in the future, with respect to some specific streaming application. You decide to use a very simple model, for instance, an $M/M/1/W$ queue (representing a whole network by just a queue is a frequent simplification, e.g. [2, 3, 8]). This model should allow you to derive standard performance measures such as the loss probability (assuming the system in equilibrium). The inputs are the arrival rate (or offered mean traffic) λ, the equivalent transmission rate μ that the whole network offers to the packets and the maximum number of packets that the network can hold, represented globally

here by the storage capacity W of the queue. This implies that these parameters can be estimated from the network characteristics and from the expected behavior of the traffic. In symbols, the loss probability p_L is given by $p_L = \varrho^W(1-\varrho)/(1-\varrho^{W+1})$ with $\varrho = \lambda/\mu \neq 1$ (if $\varrho = 1$ then $p_L = 1/(W+1)$). Since we know that the quality of the stream strongly depends on the losses, computing p_L should allow the designer to tune the appropriate control variables in order to obtain a loss probability less than some specific threshold. From a practical point of view, the problem with this approach is that the user is in fact sensitive only to his/her perception of the quality, which is a strongly subjective concept. This perceived quality depends on many factors other than the loss rate (probably some of them even not well identified), and this dependency is complex and not well understood. The approach we present here is to combine the method proposed in [19] with standard modeling techniques in order to choose, say, ϱ and W such that *the perceived quality itself* is high enough.

Let us say it again in a more formal way. The standard modeling approach in usual performance evaluation activities gives you performance metrics such as loss rates, mean delays, etc. as functions of available input data or directly controllable parameters (such as arrival rates, service rates, routing probabilities, storage capacities, etc.). Denote by $\nu_1, \nu_2, \ldots$ these data variables or parameters. If $x_1, x_2, \ldots$ represent the performance metrics, queuing models typically give to you expressions of the form $x_i = g_i(\nu_1, \nu_2, \ldots)$. For instance, in the example of the $M/M/1/W$ model, we could have the loss rate and the average delay as functions of the load ϱ and the storage capacity W. Our proposal here is to build a function giving to you the quality Q (as perceived by the user) as a function of $\nu_1, \nu_2, \ldots$ through

$$Q = f(g_1(\nu_1, \nu_2, \ldots), g_2(\nu_1, \nu_2, \ldots), \ldots). \quad (1)$$

Later in the paper we provide a closed form of such a function in the case of an audio application (see relation (5)).

Consider another situation. You need to analyze the effect of the offset of a FEC (Forward Error Correction) technique such as the one proposed in [8] on quality using a model (as, for instance, in [2, 8]). What people usually do is to build a model (some queuing system capturing the behavior of the network), with perhaps an appropriate loss model, and then, to obtain a performance metric as a function of specific input parameters. At this point, some *utility functions* are used to take into account the way users react. These functions are mappings between the performance metrics and some abstract concept of quality. Our proposal eliminates the need for these functions by providing a *validated* mapping between network parameters (and also source parameters) and perceived quality. Moreover, recall that our functions can have more than one variable. For instance, in the example proposed later in this text, we will provide a function mapping the load ϱ, the network memory capacity W, the offset of the FEC data, the type of codec and the packetization interval, into quality, for a voice stream. Therefore, there is no need for using these auxiliary utility functions: if your problem is to analyze the effects of the FEC offset (for instance, to look at its optimal value under some specific conditions) our method gives you a function $Q = f(\ldots, x, \ldots)$ as explained before, where among the different input parameters x is the FEC offset.

A last word about the modeling part of the process. In this paper we propose an approach and we chose a very simple model, an $M/M/1/W$ queue, to illustrate it because it is a well known model, and it is simple enough to allow us to keep the focus on the method. However, it should be clear that the model to choose depends on the accuracy you need, the data you have, the questions you have to answer, the available computer power if a computer is needed, etc. For instance, if you have to estimate the performance of some UDP-based application using a TCP-friendly control algorithm and you decide to use some specific model to capture the aspect of the system behavior you are interested in, you have to build your $f()$ function in order to be able to perform the proposed mapping illustrated in (1). If your model is accurate (that is, if the quality of your $g_i()$ functions is high, then (1) will give you a good estimate of the ultimate quality, directly as a function of your data. Coming back to the first simple example used here, perhaps you will decide to represent your system by a complex Markovian queuing network having many "low level" input parameters (instead of just ϱ and W), to be solved only numerically. In that case, $f()$ will be accessible through a numerical procedure, but again, the result will be an accurate estimation of the quality as seen by the end–user of the application, whose efficiency will be essentially proportional to the efficiency of the model you use.

The rest of the paper is organized as follows. Section 2 presents the quality assessment method used. In section 3, we show how to use it together with classical queuing models. Section 3 also presents an example of the results obtained with our method on one–way VoIP streams. Finally, section 4 presents our conclusions and future work in this area.

2. Pseudo–subjective Quality Assessment

Correctly assessing the perceived quality of a multimedia stream is not an easy task. As quality is, in this context, a very subjective concept, the best way to evaluate it is to have real people do the assessment. There exist standard methods for conducting *subjective* quality evaluations, such as the ITU-P.800 [17] recommendation for telephony, or the ITU-R BT.500-10 [15] for video. The main problem with subjective evaluations is that they are very expensive (in terms of both time and manpower) to carry out, which makes them hard to repeat often. And, of course, they cannot be a part of an automatic process.

Given that subjective assessment is expensive and impractical, a significant research effort has been done in order to obtain similar evaluations by *objective* methods, i.e., algorithms and formulas that measure, in a certain way, the quality of a stream. The most commonly used objective measures for speech/audio are Signal-to-Noise Ratio (SNR), Segmental SNR (SNRseg), Perceptual Speech Quality Measure (PSQM) [6], Measuring Normalizing Blocks (MNB) [28], ITU E–model [16], Enhanced

Modified Bark Spectral Distortion (EMBSD) [30], Perceptual Analysis Measurement System (PAMS) [23] and PSQM+ [5]. For video, some examples are the ITS' Video Quality Metric (VQM) [4, 27], EPFL's Moving Picture Quality Metric (MPQM), Color Moving Picture Quality Metric (CMPQM) [25, 26], and Normalization Video Fidelity Metric (NVFM) [26]. As stated in the Introduction, these quality metrics often provide assessments that do not correlate well with human perception, and thus their use as a replacement of subjective tests is limited. Except for the ITU E–model, all these metrics propose different ways to compare the received sample *with the original one*. The E–model allows to obtain an approximation of the perceived quality as a function of several ambient, coding and network parameters, to be used for network capacity planning. However, as stated in [13] and even in its specification [16], its results do not correlate well with subjective assessments either.

The method used here [22, 19] is an hybrid between subjective and objective evaluation. The idea is to have several distorted samples evaluated subjectively, and then use the results of this evaluation to teach a RNN the relation between the parameters that cause the distortion and the perceived quality. In order for it to work, we need to consider a set of P parameters (selected *a priori*) which may have an effect on the perceived quality. For example, we can select the codec used, the packet loss rate of the network, the end–to–end delay and/or jitter, etc. Let this set be $\mathcal{P} = \{\pi_1, \ldots, \pi_P\}$. Once these *quality–affecting* parameters are defined, it is necessary to choose a set of representative values for each π_i, with minimal value $\pi_{\min}$ and maximal value $\pi_{\max}$, according to the conditions under which we expect the system to work. Let $\{p_{i1}, \cdots, p_{iH_i}\}$ be this set of values, with $\pi_{\min} = p_{i1}$ and $\pi_{\max} = p_{iH_i}$. The number of values to choose for each parameter depends on the size of the chosen interval, and on the desired precision. For example, if we consider the packet loss rate as one of the parameters, and if we expect its values to range mainly from 0% to 5%, we could use 0, 1, 2, 5 and perhaps also 10% as the selected values. In this context, we call *configuration* a set with the form $\gamma = \{v_1, \ldots, v_P\}$, where v_i is one of the chosen values for p_i.

The total number of possible configurations (that is, the number $\prod_{i=1}^{P} H_i$) is usually very large. For this reason, the next step is to select a subset of the possible configurations to be subjectively evaluated. This selection may be done randomly, but it is important to cover the points near the boundaries of the configuration space. It is also advisable not to use a uniform distribution, but to sample more points in the regions near the configurations which are most likely to happen during normal use. Once the configurations have been chosen, we need to generate a set of "distorted samples", that is, samples resulting from the transmission of the original media over the network under the different configurations. For this, we use a testbed, or network simulator. For instance, in the case study presented in section 3, we've used a proxy to generate the desired loss rates and distributions (cf. section 3 for details) on a local area network.

Formally, we must select a set of M media samples (σ_m), $m = 1, \cdots, M$, for instance, M short pieces of audio (subjective testing standards advise to use sequences having an average 10 sec length). We also need a set of S configurations denoted by $\{\gamma_1, \cdots, \gamma_S\}$ where $\gamma_s = (v_{s1}, \cdots, v_{sP})$, v_{sp} being the value of parameter π_p in configuration γ_s. From each sample σ_i, we build a set $\{\sigma_{i1}, \cdots, \sigma_{iS}\}$ of samples that have encountered varied conditions when transmitted over the network. That is, sequence σ_{is} is the sequence that arrived at the receiver when the sender sent σ_i through the source-network system where the P chosen parameters had the values of configuration γ_s.

Once the distorted samples are generated, a subjective test [15, 17] is carried out on each received piece σ_{is}. After statistical processing of the answers, the sequence σ_{is} receives the value μ_{is} (often, this is a *Mean Opinion Score*, or MOS). The idea is then to associate each configuration γ_s with the value

$$\mu_s = \frac{1}{M} \sum_{m=1}^{M} \mu_{ms}.$$

At this step we have a set of S configurations $\gamma_1, \ldots, \gamma_S$. Configuration s has value μ_s associated with it. We randomly choose S_1 configurations among the S available. These, together with their values, constitute the "Training Database". The remaining $S_2 = S - S_1$ configurations and their respective values constitute the "Validation Database", reserved for further (and critical) use in the last step of the process.

The next step is to train a statistical learning tool (in our case, a RNN) to learn the mapping between configurations and values as defined by the Training Database. Assume that the selected parameters have values scaled into [0,1] and the same with quality. Once the tool "captured" the mapping, that is, once the tool trained, we have a function $f()$ from $[0,1]^P$ into $[0,1]$ mapping now any possible value of the (scaled) parameters into the (also scaled) quality metric. The last step is the validation phase: we compare the value given by $f()$ at the point corresponding to each configuration γ_s in the Validation Database to μ_s; if they are close enough for all of them, the RNN is validated (in Neural Network Theory, we say that the tool *generalizes well*). If the RNN did not validate, it would be necessary to review the chosen architecture and configurations. In the studies we have conducted while developing this approach, not once did the RNN fail to be validated. In fact, the results produced by the RNN are generally closer to the MOS than that of the human subjects (that is, the error is less than the average deviation between human evaluations). As the RNN generalizes well, it suffices to train it with a small (but well chosen) part of the configuration space, and it will be able to produce good assessments for any configuration in that space. The choice of the RNN as an approximator is not arbitrary. We have experimented with other tools, namely Artificial Neural Networks and Bayesian classifiers, and found that RNN perform better in the context considered. ANN exhibited some performance problems due to overtraining, which we did not find when using RNN. As for the Bayesian classifier, we found that while it worked, it did so quite roughly, with much less precision than RNN. Besides, it is only able to provide discrete quality scores, while the NN approach allows for a finer view of the quality function.

The neural network model used has some interesting mathematical properties, which allow, for example, to obtain the derivatives of the output with respect to any of the inputs, which is useful for evaluating the performance of the network under changing conditions (see next section). Besides, we have seen that a well trained RNN will be able to give reasonable results even for parameter values outside the ranges considered during training, i.e. it extrapolates well.

The method proposed produces good evaluations for a wide range variation of all the quality affecting parameters, at the cost of one subjective test. In [19], the authors present results which have a correlation coefficient of about 0.97 with human results for video streams. For a detailed comparison of this approach with other objective audio assessment metrics from a performance standpoint, see [21].

2.1. RNN: Open Queuing Networks as Statistical Learning Tools

Let us briefly describe the way we can use a specific class of queuing networks as a very efficient statistical learning tool. The mathematical object and its use in learning was introduced and developed in [9, 11, 12].

An RNN is an open Markovian queuing network with positive and negative customers, also called a G-network. We have N nodes (or neurons) which are $./M/1$ queues (the service rate of node i is denoted by r_i), interconnected, receiving customers from outside and sending customers out of the network. Customers are "positive" or "negative"; the arrival flow of positive (respectively negative) customers arriving at node i from outside is Poisson with rate λ_i^+ (respectively λ_i^-). After leaving neuron (queue) i, a customer leaves the network with probability d_i, goes to queue j as a positive customer with probability p_{ij}^+ and as a negative customer with probability p_{ij}^-. When a negative customer arrives at a node i (either from outside or from another queue) it disappears, removing the last customer at i, if any. Transfers between queues are, as usual with queuing network models, instantaneous. This means that negative customers can not be observed; at any point in time there are only positive customers in the network; negative customers act only as *signals*, modifying the behavior of the system.

Let us denote by N_t^i the number of customers in queue i at time t. Then, it was proved in [10, 11] that when the (Markov) process $\vec{N}_t = (N_t^1, \cdots, N_t^M)$ is stable, its stationary distribution is of the product-form type: that is, assuming that $(\vec{N}_t)$ is stationary, we have

$$\Pr(\vec{N}_t = (k_1, \cdots, k_M)) = \prod_{i=1}^{M} (1-\rho_i)\rho_i^{k_i}.$$

The factors $\rho_1, \cdots, \rho_M$ in this expression are the loads of the nodes in the network. The specificities of these networks make that these loads are not obtained by solving a linear system (as in the Jackson case) but by solving the following non-linear one:

$$\rho_i = \frac{\lambda_i^+ + \sum_{j=1}^{N} \rho_j r_j p_{ji}^+}{r_i + \lambda_i^- + \sum_{k=1}^{N} \rho_k r_k p_{ki}^-}.$$

It can then be proved that when this system has a solution $\rho_1, \cdots, \rho_M$ such that for each node i it is $\rho_i < 1$, then the process is stable, and the product-form result holds (see [10]).

To use such a queuing network as a learning tool, we perform the following mapping: the input variables (bit rate, loss rate, etc.) are scaled into [0,1] and then associated with the external arrival rates of positive customers at P specific nodes of the network $\lambda_1^+, \cdots, \lambda_P^+$. The remaining external arrival rates of positive customers are set to 0; we also set to zero the external rates of negative customers. The quality of the sequence after normalization also on [0,1] is mapped to the load of a specific node o in the system. The problem now is to find a network such that when $\lambda_1^+ = v_{1s}, \cdots, \lambda_P^+ = v_{Ps}$, then the load of the chosen node o is close to μ_s. This is an optimization problem where the control variables are now the remaining parameters of the network: the service rates r_i and the routing probabilities p_{ij}^+ and p_{ij}^-.

For all neurons i such that $d_i < 1$ (that is, for all neuron that does not send all its signals (its customers) out of the network), we denote $w_{ij}^+ = r_i p_{ij}^+$ and $w_{ij}^- = \mu_i p_{ij}^-$. These w_*^* factors are called *weights* as in the standard neural network terminology, and they play a similar role in this model. Instead of optimizing with respect to the service rates and the transition probabilities, the standard approach is to do it with respect to the weights, and just to keep constant the service rates of the "output" neurons (those neurons j where $d_j = 1$).

The optimization problem can be solved using standard techniques such as gradient descent (observe that we are able to compute any partial derivative of the output, using the non-linear system of equations satisfied by the occupation rates). See the given papers on learning with RNN for details about the available procedures.

3. Our approach at work

In this section we describe in detail the application of the approach we propose to a specific type of situation and we provide some numerical results to illustrate the method and its use. Our choice is thus to study the performance of a one–way voice stream transmitted over a best–effort network.

3.1. Modeling

We resume here the main example discussed in the Introduction. That is, we want to predict the behavior of some audio streaming application over an IP network (for instance, a low bandwidth radio). As discussed, our approach has two parts: a

modeling part for performance evaluation purposes, and a perceived quality evaluation function that will receive inputs from the former. For illustration purposes, and in order to simplify the presentation, let us consider the whole network represented by an $M/M/1/W$ model as discussed above. The input parameters of the model are its load ϱ and its storage capacity W. We also want to take into account two different codecs, the possibility of using FEC and, in that case, its offset, and the packetization interval (i.e, the length in milliseconds of speech contained in each packet). These three parameters are specific to the source. We will consider that, as is the case in the current Internet, audio traffic represents a very small fraction of the total traffic, and therefore the impact of changing codecs, or using more or less redundancy in the flow is negligible on the global load ϱ. From the modeling point of view, we will just focus on losses. Of course, more parameters can be taken into account, as long as they can be derived from the chosen model. For example, if we were to consider interactive applications as opposed to one-way streaming, network delay and jitter would need to be considered too.

As recalled before, we know that the loss probability or loss rate is related to the inputs variables ϱ and W through the expression $p_L = \varrho^W(1-\varrho)/(1-\varrho^{W+1})$ if $\varrho \neq 1$, and $p_L = 1/(W+1)$ if $\varrho = 1$. The loss rate alone is too poor to capture the way losses have an impact on quality. This is largely supported in the literature [7, 14, 20], and is confirmed in our experiences. In many cases, the perceived qualities of different samples are quite different even if the loss rate in the network is the same. For this reason, we chose to use a second parameter to characterize the loss process: the mean loss burst size (denoted herein by mlbs).

This means that our quality evaluation component is a function $f()$ of six variables: the four source-related (codec, existence of FEC, FEC offset, packetization interval) and two characterizing the network effect: loss rate (p_L) and mean loss burst size (mlbs).

In our model choice, we need to compute mlbs as a function of ϱ and W, which is a simple task. The probability that a burst of losses has size j, $j \geq 1$, is equal to $p^{j-1}q$ where $p = \varrho/(1+\varrho)$ and $q = 1-p = 1/(1+\varrho)$. This is because a burst has size j iff after a first loss, the *next* packet is lost (probability p), and the next one, and so on exactly $j-1$ times, and the following packet is not lost, which happens with probability q. The expectation mlbs is then $\mathrm{mlbs} = \sum_{j\geq 1} p^{j-1}q = 1/q = 1+\varrho$.

Recall now that we choose to limit the analysis to some specific ranges according to the goals of the study. For the two network parameters, assume we decide that the loss rate p_L will be considered in the range $[0, p_0]$ (for instance, $p_0 = 0.15$, because with higher loss rates most current VoIP applications see their quality degraded way below acceptable levels). In the same way, we consider $\mathrm{mlbs} \in [0, \mathrm{mlbs}_0]$ (for instance, $\mathrm{mlbs}_0 = 4$ following the observation of traces in the Internet [29]). Of course, this must be translated into our final input parameters, which for the network part of the analysis are ϱ and W. Since

$$\mathrm{mlbs} = 1 + \varrho,$$

we immediately have the constraint

$$\varrho \leq \varrho_0 = \mathrm{mlbs}_0 - 1. \tag{2}$$

The discussion on the loss probability is a little bit more complex. Since the use of the $M/M/1/W$ model is illustrative here, let us just give the main element in the simplest case of $\mathrm{mlbs}_0 \leq 2$. In this case, $\varrho \leq \varrho_0 = 1$. We want that

$$\frac{1-\varrho}{1-\varrho^{W+1}}\varrho^W \leq p_0,$$

which after some algebra gives, for any $\varrho < 1$,

$$W \geq W_0 = \left\lceil \ln\left(\frac{p_0}{1-\varrho(1-p_0)}\right) / \ln(\varrho) \right\rceil. \tag{3}$$

If $\varrho = 1$, then $p_L = (W+1)^{-1}$ and

$$p_L \leq p_0 \Longleftrightarrow W \geq W_0 = \left\lceil \frac{1-p_0}{p_0} \right\rceil.$$

This means that our analysis will consider the values of ϱ and W in specific domains, which are also meaningful data for the network designer.

3.2. The quality evaluator

Let us consider now the quality evaluator part. We have already said that we have chosen our network parameters so that they reflect conditions that are actually found on the Internet [29]. As we considered only unidirectional streams, the most relevant network parameters were the loss rate and the mean loss burst size. As seen in section 3.1, those parameters are easily derived from our network model. If need be (for instance, in order to evaluate a two–way stream), other network parameters such as the mean end–to–end delay and jitter could be derived and used.

As for the coding parameters, we chose the codec, the redundancy scheme used (whether FEC was used, and if that was the case, the offset of the FEC packets), and the packetization interval (i.e, the length in milliseconds of speech contained in each packet).

In order to transport our streams, we used the Robust Audio Tool (RAT) [18], which was developed as part of the MICE project at UCL. This conferencing tool provides all the needed configuration options, and it is easily scriptable, which greatly simplifies the creation of the distorted samples.

We implemented our network testbed on a LAN, on which we generated losses according to a simplified Gilbert model consisting of a two–state homogeneous Markov chain [29]. This model has been shown to produce loss processes very similar to those found on the Internet [1, 24, 29]. Working on a LAN, where losses are negligible, allowed us to finely control the loss process during the generation of the samples. Table 1 presents a summary of the parameters used.

Parameter	Values
Loss rate	0%...15%
Mean loss burst size	1...2.5
Codec	PCM Linear 16 bits, GSM
FEC	ON/OFF
FEC offset	1...3
Packetization interval	20, 40, and 80ms

Table 1. Network and encoding parameters and values used

With twelve original samples, we chose 112 parameter configurations (with a bias toward PCM encoding and small packet sizes, and making sure that border configurations were present). We then generated 112 groups of four distorted samples each. We had these samples evaluated by 17 people, and after performing the statistical screening recommended in [17], we had to discard the evaluations of one of the subjects. The 112 configurations were divided into two groups, one of 92 configurations used to train the RNN (so, for the Training Database), and another one of 20, used as a control group (for the Validation Database). The configurations used for training were chosen randomly among the whole set.

Once we had the subjective evaluation, we proceeded to train several RNN, in order to find the most appropriate architecture to use. In [19], the authors suggest a 3–layer feed-forward architecture, with a large hidden layer. We used 3 different architectures, namely:

- a 3–layer feed-forward RNN as proposed in [19],
- a 3–layer RNN, with a recurrent hidden layer, and
- a very simple, 2–layer feed-forward RNN, with the inputs connected directly to the output neuron (see Figure 1).

Surprisingly, we didn't find any significant difference in the performances of the three architectures for our application. We tried using a bootstrap–like approach to train the smallest network with the output of the recurrent one (this allows to build a large Training Database), but there was no noticeable difference in their outputs either. We find this very interesting, since it allows to represent the quality of the speech stream with a relatively simple formula, obtained by substituting our coefficients in the RNN general formula. Figure 1 shows the structure of our RNN, and Figure 2 shows its performance for the Validation Database (i.e. a plot of the estimated MOS values versus the actual MOS values).

If we look at the topology depicted in Figure 1, we see that it is of the feed-forward type (there is no circuit in it, or, in other words, every customer visits a given node at most only once). In this case, the calculations are particularly simple: denote by 1 to n the entry neurons and by $n+1$ the only output one. We have that for all $i = 1, \cdots, n$,

$$\rho_i = \frac{\lambda_i^+}{r_i} = \frac{\lambda_i^+}{w_{i,n+1}^+ + w_{i,n+1}^-},$$

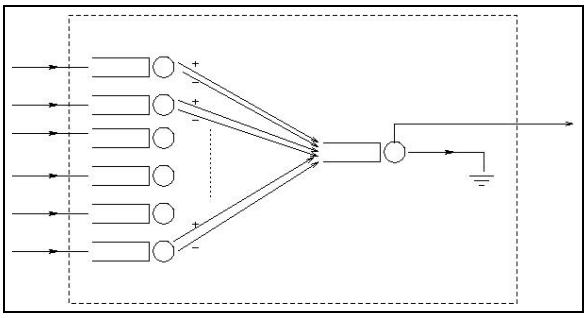

Figure 1. The RNN architecture used for this case study: a very simple 2–layer feed-forward network

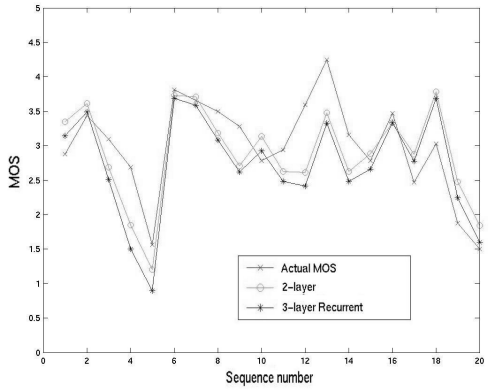

Figure 2. MOS for the 20 control samples: Humans (Actual MOS) vs. 3–layer recurrent RNN vs. 2–layer feed forward RNN

and

$$\rho_{n+1} = \frac{\sum_{j=1}^{n} \rho_j w_{j,n+1}^+}{r_{n+1} + \sum_{j=1}^{n} \rho_j w_{j,n+1}^-}.$$

In our experiments, where $n = 6$, we decided to keep constant the rate of neuron 7 (the value we used was $r_7 = 0.01$). Once trained, the values of the weights are given in Tables 2 and 3.

Observe that the preceding discussion leads to the following expression for the quality of the stream:

$$Q = \rho_7 = \frac{\sum_{i=1}^{6} a_i \lambda_i^+}{0.01 + \sum_{i=1}^{6} b_i \lambda_i^+} \tag{4}$$

where

$$a_i = \frac{w^+_{i,7}}{w^+_{i,7} + w^-_{i,7}} \quad \text{and} \quad b_i = \frac{w^-_{i,7}}{w^+_{i,7} + w^-_{i,7}}.$$

Such a simple and explicit expression allows to easily analyze the variation of the quality with respect to specific parameters. Formally, setting $a_0 = 0$ and $b_0 = 0.01$,

$$\frac{\partial Q}{\partial \lambda^+_k} = \frac{c_0 + \sum_{i=1}^{6} c_i \lambda^+_i}{(b_0 + \sum_{i=1}^{6} b_i \lambda^+_i)^2}$$

where

$$c_i = \begin{vmatrix} a_k & a_i \\ b_k & b_i \end{vmatrix} = a_k b_i - b_k a_i \ \text{(in particular, } c_k = 0).$$

For example, in Figure 8 we show the sensitivity of the quality with respect to the load for two values of the capacity, in a specific configuration (see below).

Let us now give an example of a closed form expression for quality, as a function of the parameters ϱ and W (that is, an instance of relation (1)). In Table 1 we have the list of the 6 entries in the neural network (that is, in function $f()$ using the notation given in the Introduction). Respecting the same ordering, x_1 is the loss rate, going from 0 to 0.15, x_2 is the mean loss burst size, going from 0.4 to 1.0 (recall that we scale to work with variables in $[0, 1]$, 0.4 coming from $1/2.5$), etc. The RNN tool provides us with the function $f(x_1, \cdots, x_6)$ mapping these 6 variables into a MOS quality metric. Assume that we fix variables x_3 to x_6 to some specific values, and that we build a new function as in relation (1) having as input variables $x_3, \cdots, x_6$, plus W (the buffer size) and ϱ, the *global* load of the link. To this purpose, we use the expressions of x_1 (the loss probability), and x_2 (the normalized mean burst loss rate), as a function of these two new input variables W and ϱ, provided by the analysis of the associated performance model (the $M/M/1/W$ queue in our example). Recall that the limited range of the loss rate and the mean loss burst size lead to corresponding specific ranges to the load and the buffer size, namely $\varrho \leq \varrho_0$ and $W \geq W_0$ (see relations (2) and (3)). Function $f(x_1, \cdots, x_6)$ is rational with known coefficients (as previously described). Once variables x_3 to x_6 fixed to specific values, and x_1 (resp. x_2) replaced by $x_1 = (1 - \varrho)\varrho^W(1 - \varrho^{-(W+1)})$ (resp. $x_2 = 1 + \varrho$), we obtain, after some algebra,

$$Q = \frac{\alpha + \beta\varrho + \gamma\varrho^W - (\gamma + \alpha)\varrho^{W+1} - \beta\varrho^{W+2}}{\alpha' + \beta'\varrho + \gamma'\varrho^W - (\gamma' + \alpha')\varrho^{W+1} - \beta'\varrho^{W+2}}. \quad (5)$$

The parameters α, β, γ, α', β' and γ' are functions of the encoding ones. As an example, assume we chose the PCM codec, FEC with an offset of 1 and a packetization interval equal to 20 ms. We obtain $\alpha = 0.1326$, $\beta = 0.0201$, $\gamma = 0.0674$, $\alpha' = 0.1659$, $\beta' = 0.0399$, $\gamma' = 0.9326$. The quality function becomes (after multiplying numerator and denominator by 100 for typographical purposes)

$$Q = \frac{13.26 + 2.01\varrho + 6.74\varrho^W - 20\varrho^{W+1} - 2.01\varrho^{W+2}}{16.59 + 3.99\varrho + 93.26\varrho^W - 109.8\varrho^{W+1} - 3.99\varrho^{W+2}}.$$

Parameter	$w^+_{i,7}$
Codec	0.831879
FEC	1.53147
FEC Offset	1.08491
Loss rate	0.193885
Mean burst size	1.12165
Packetization interval	1.50425

Table 2. Weights of positive connexions in the expression of quality (see relation (4))

Parameter	$w^-_{i,7}$
Codec	1.50221
FEC	1.64266
FEC Offset	1.36289
Loss rate	2.68204
Mean burst size	2.23472
Packetization interval	1.59028

Table 3. Weights of negative connexions in the expression of quality (see relation (4))

Figures 3 through 8 illustrate the kind of insight that a network designer can gain from using the approach proposed to know how quality reacts to network impairments. Currently, networks tend to be over–dimensioned if some level of performance is to be attained. With the method we provide, it should be possible to adjust the network more finely, obtaining the desired QoS levels without wasting resources. For example, knowing (approximately) the expected traffic levels, and the fraction of that traffic which is expected to be voice flows, one can adjust the needed capacity (and thus the expected load), so as to maintain voice quality over a certain threshold. Moreover, one could even evaluate the utility (or lack thereof) of using FEC to protect those flows, depending on the expected increase in load that it would generate, and its impact on quality. The same idea is applicable to other types of real–time traffic, such as video, for example.

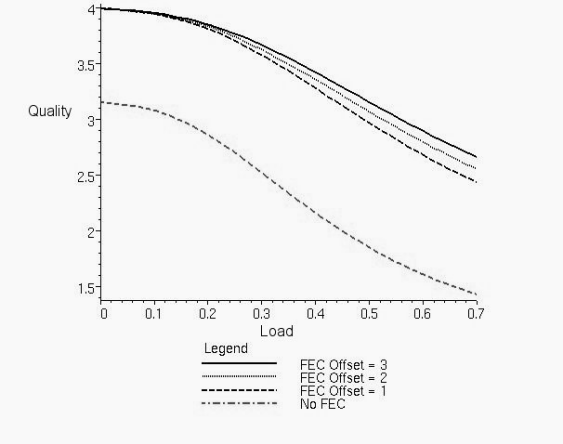

Figure 3. Quality as a function of the load, for different FEC settings. W = 3

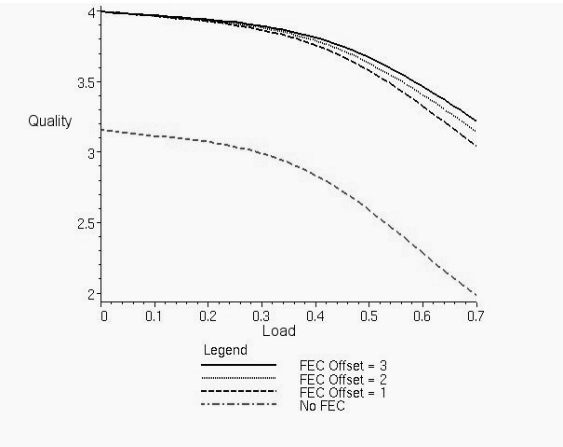

Figure 4. Quality as a function of the load, for different FEC settings. W = 5

In Figures 3 and 4, we can see how different FEC settings provide different qualities under an increasing load, for two values of network capacity. We can see that the efficiency of the FEC varies with W, and also what's the gain obtained. A quality (MOS) value of 3 is considered to be "acceptable" [17]. It is easy to see that an increase of 2 in W allows to have an acceptable quality with a quite higher load. not only that (which is to be expected, anyway), but we can know how much higher, and then make a design decision based on the QoS levels we want to attain, and the costs associated with the higher capacity. It is also interesting to see that this kind of curve is a more accurate representation of the FEC performance than the utility functions used for example in [3, 8], which had to be designed artificially.

Figures 5 and 6 present the perceived quality as a function of

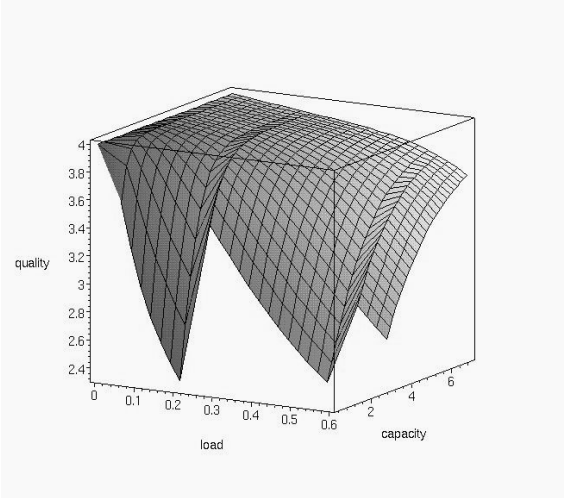

Figure 5. Quality as a function of the load and network capacity, with FEC (offset = 1).

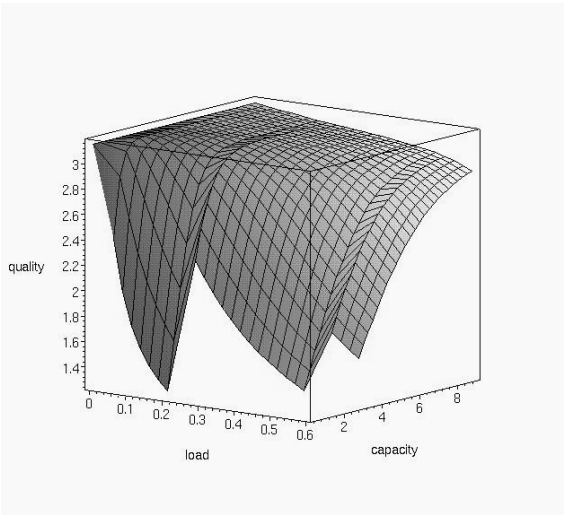

Figure 6. Quality as a function of the load and network capacity, without FEC

both network load and capacity, with and without FEC, respectively. (ϱ, W) plane is feasible, since the pair (ϱ, W) must satisfy the restriction given in relations such as (3). see that when not using FEC, the variations in quality as the load increases are more pronounced than when a FEC scheme is present. This kind of plot allows to measure the benefits of using FEC or not, depending on what the expected amount of VoIP traffic is in our network. For example, if voice traffic is predominant in the network, and we know that using FEC increases the global load by a factor x, we can assess whether it'll be useful or not to enable it, or even if we could get away with a smaller value of W and no FEC (this doesn't seem to be the case here, but it may

very well be the case for other kind of application).

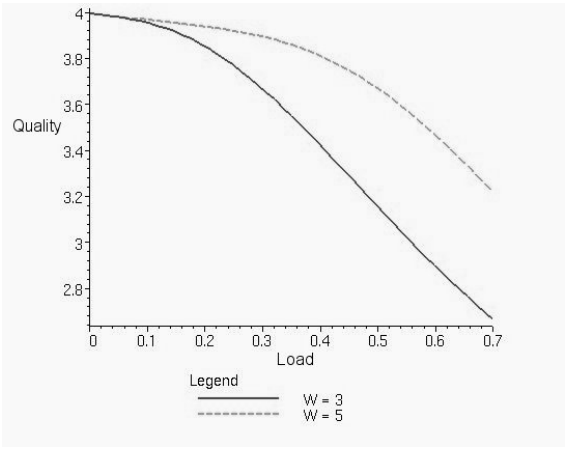

Figure 7. Quality as a function of network load, for two values of W, and with FEC (offset = 1).

Figure 7 shows two "cuts" of figure 5, which show the quality as a function of load for two values of W.

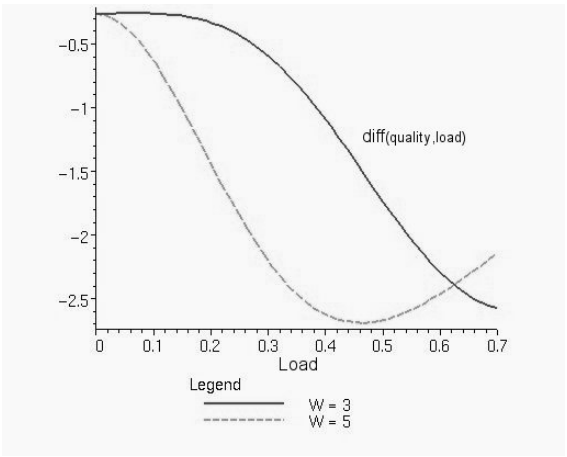

Figure 8. $\partial Q/\partial \varrho$ as function of ϱ

Finally, figure 8 illustrates the differences in the variation of quality under the load, as the derivative of the quality with respect to the load, for two different network capacities.

4. Conclusions

In this paper, we present a new approach to evaluating network performance, by taking into account the end–user perception of the quality of networked multimedia applications. We do this by integrating classic modeling techniques with a pseudo–subjective multimedia quality assessment approach. This allows us to predict quality metrics based on the expected network conditions very accurately.

The method described in this paper allows to estimate the quality as a function of parameters typical of network design. In the example used to illustrate the approach, these are load and capacity. However, the technique basically works with any network model, provided that it allows to derive the relevant performance metrics. An important feature of our proposal is that the performance model and the pseudo-subjective evaluation tool must fit together. Typically, the starting point is to build the pseudo-subjective evaluation tool, using the most relevant source and network parameters for the applications considered. Then, the model used must provide the network parameters' values needed by the former. In the example used in the paper to explain our approach, the selected quality affecting network parameters were the loss rate and the mean loss burst size. The model (a simple $M/M/1/W$ queue) was then used to provide them as functions of the load and the capacity represented by W.

We believe that this method will allow network designers to better take multimedia applications into account when dimensioning new networks, allowing them to predict the expected qualities for each application, by coupling a model of the network with quality predictors adapted to each application. As this kind of applications becomes more and more common, we believe that the usefulness of being able to predict how they will work on a new network will becomes more and more evident.

References

[1] B. Ahlgren, A. Andersson, O. Hagsand, and I. Marsh. Dimensioning links for IP telephony. Technical Report T2000–09, Swedish Institute of Computer Science (SICS), 2000.

[2] E. Altman, C. Barakat, and V. M. R. R. On the utility of FEC mechanisms for audio applications. *Lecture Notes in Computer Science*, 2156, 2001.

[3] E. Altman, C. Barakat, and V. M. R. R. Queueing analysis of simple FEC schemes for IP telephony. In *Proceedings of INFOCOM '01*, pages 796–804, 2001.

[4] W. Ashmawi, R. Guerin, S. Wolf, and M. Pinson. On the impact of policing and rate guarantees in diff-serv networks: A video streaming application perspective. *Proceedings of ACM SIGCOMM'2001*, pages 83–95, 2001.

[5] J. Beerends. Improvement of the p.861 perceptual speech quality measure. ITU-T SG12 COM-34E, Dec. 1997.

[6] J. Beerends and J. Stemerdink. A perceptual speech quality measure based on a psychoacoustic sound representation. *Journal of Audio Eng. Soc.*, 42:115–123, Dec. 1994.

[7] J.-C. Bolot, S. Fosse-Parisis, and D. F. Towsley. Adaptive FEC-based error control for internet telephony. In *Proceedings of INFOCOM '99*, pages 1453–1460, 1999.

[8] J.-C. Bolot and A. Vega Garcia. The case for FEC-based error control for packet audio in the internet. In *ACM Multimedia Systems*, 1996.

[9] E. Gelenbe. Random neural networks with negative and positive signals and product form solution. *Neural Computation*, 1(4):502–511, 1989.

[10] E. Gelenbe. Stability of the random neural network model. In *Proc. of Neural Computation Workshop*, pages 56–68, Berlin, West Germany, Feb. 1990.

[11] E. Gelenbe. G-networks: new queueing models with additional control capabilities. In *Proceedings of the 1995 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pages 58–59, Ottawa, Ontario, Canada, 1995.

[12] E. Gelenbe and K. Hussain. Learning in the multiple class random neural network. *IEEE Trans. on Neural Networks*, 13(6):1257–1267, 2002.

[13] T. A. Hall. Objective speech quality measures for internet telephony. In *Voice over IP (VoIP) Technology, Proceedings of SPIE*, volume 4522, pages 128–136, Denver, CO, USA, Aug. 2001.

[14] D. Hands and M. Wilkins. A study of the impact of network loss and burst size on video streaming quality and acceptability. In *Interactive Distributed Multimedia Systems and Telecommunication Services Workshop*, Oct. 1999.

[15] ITU-R Recommendation BT.500-10. Methodology for the subjective assessment of the quality of television pictures. In *International Telecommunication Union*, Mar. 2000.

[16] ITU-T Recommendation G.107. The E-model, a computational model for use in transmission planning.

[17] ITU-T Recommendation P.800. Methods for subjective determination of transmission quality.

[18] U. C. London. Robust Audio Tool website: `http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/index.html`.

[19] S. Mohamed and G. Rubino. A study of real–time packet video quality using random neural networks. *IEEE Transactions On Circuits and Systems for Video Technology*, 12(12):1071 –1083, Dec. 2002.

[20] S. Mohamed, G. Rubino, H. Afifi, and F. Cervantes. Real–time video quality assessment in packet networks: A neural network model. In *Proceedings of International Conference on Parallel and Distibuted Processing Techniques and Applications (PDPTA'01)*, Las Vegas, Nevada, USA, June 2001.

[21] S. Mohamed, G. Rubino, and M. Varela. A method for quantitative evaluation of audio quality over packet networks and its comparison with existing techniques. In *Proceedings of MESAQIN'04*, Prague, Czech Republic, June 2004.

[22] S. Mohamed, G. Rubino, and M. Varela. Performance evaluation of real-time speech through a packet network: a random neural networks-based approach. *Performance Evaluation*, 57(2):141–162, 2004.

[23] A. Rix. Advances in objective quality assessment of speech over analogue and packet-based networks. In *the IEEE Data Compression Colloquium*, London, UK, Nov. 1999.

[24] H. Sanneck, G. Carle, and R. Koodli. A framework model for packet loss metrics based on loss runlengths. In *Proceedings of the SPIA/ACM SIGMM Multimedia Computing and Networking Conference*, pages 177–187, San Jose, CA, Jan. 2000.

[25] C. van den Branden Lambrecht. Color moving picture quality metric. In *Proceedings of the IEEE International Conference on Image Processing*, Sept. 1996.

[26] C. van den Branden Lambrecht. *Perceptual Models and Architectures for Video Coding Applications*. PhD thesis, EPFL, Lausanne, Switzerland, 1996.

[27] S. Voran. The development of objective video quality measures that emulate human perception. In *IEEE GLOBECOM*, pages 1776–1781, 1991.

[28] S. Voran. Estimation of perceived speech quality using measuring normalizing blocks. In *IEEE Workshop on Speech Coding For Telecommunications Proceeding*, pages 83–84, Pocono Manor, PA, USA, Sept. 1997.

[29] M. Yajnik, S. Moon, J. Kurose, and D. Towsley. Measurement and modeling of the temporal dependence in packet loss. In *Proccedings of IEEE INFOCOM '99*, pages 345–352, 1999.

[30] W. Yang. *Enhanced Modified Bark Spectral Distortion (EMBSD): an Objective Speech Quality Measrure Based on Audible Distortion and Cognition Model*. PhD thesis, Temple University Graduate Board, May 1999.