Automatic measure of Quality of Experience with the PSQA approach

Dagstuhl, 2009

from Keynote talk
at 18th ITC
Specialist Seminar
May 2008, BTH, Karlskrona, Sweden

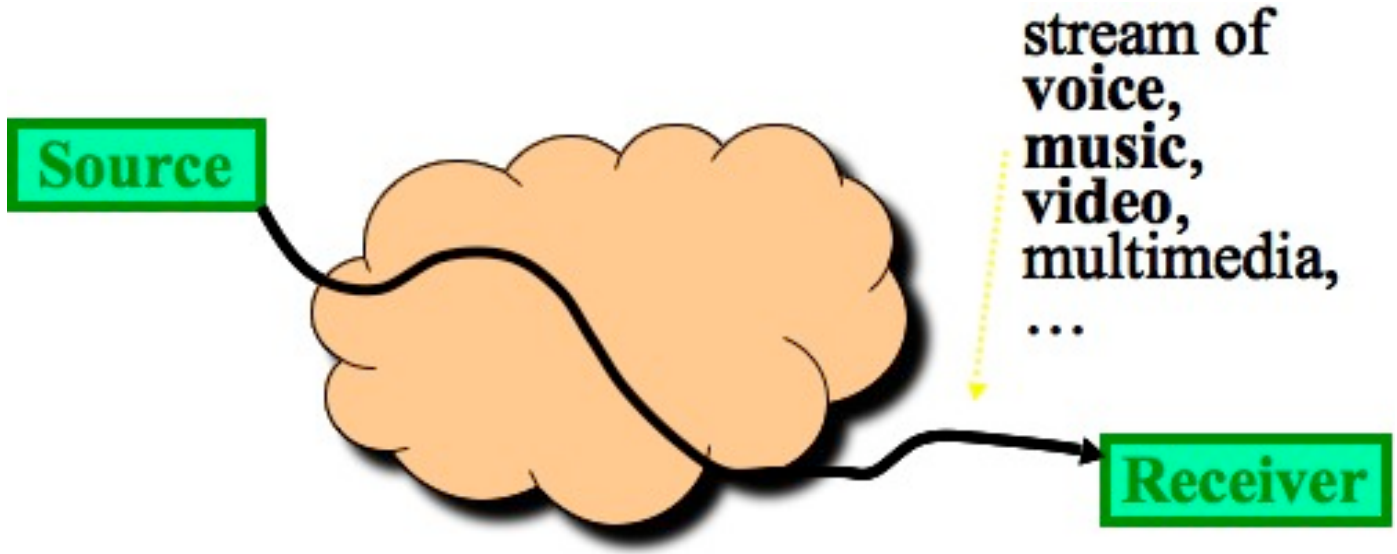
G. Rubino

INRIA Rennes, France

Outline

Main topic

Consider multimedia flows over an IP network:



Our goal is to measure quality, as perceived by the user but automatically, in real time, and using this for network analysis, design and control.
This is a network oriented approach, and the idea behind is to do this evaluation on long periods, for networking oriented goals.

Perceived quality and subjective testing

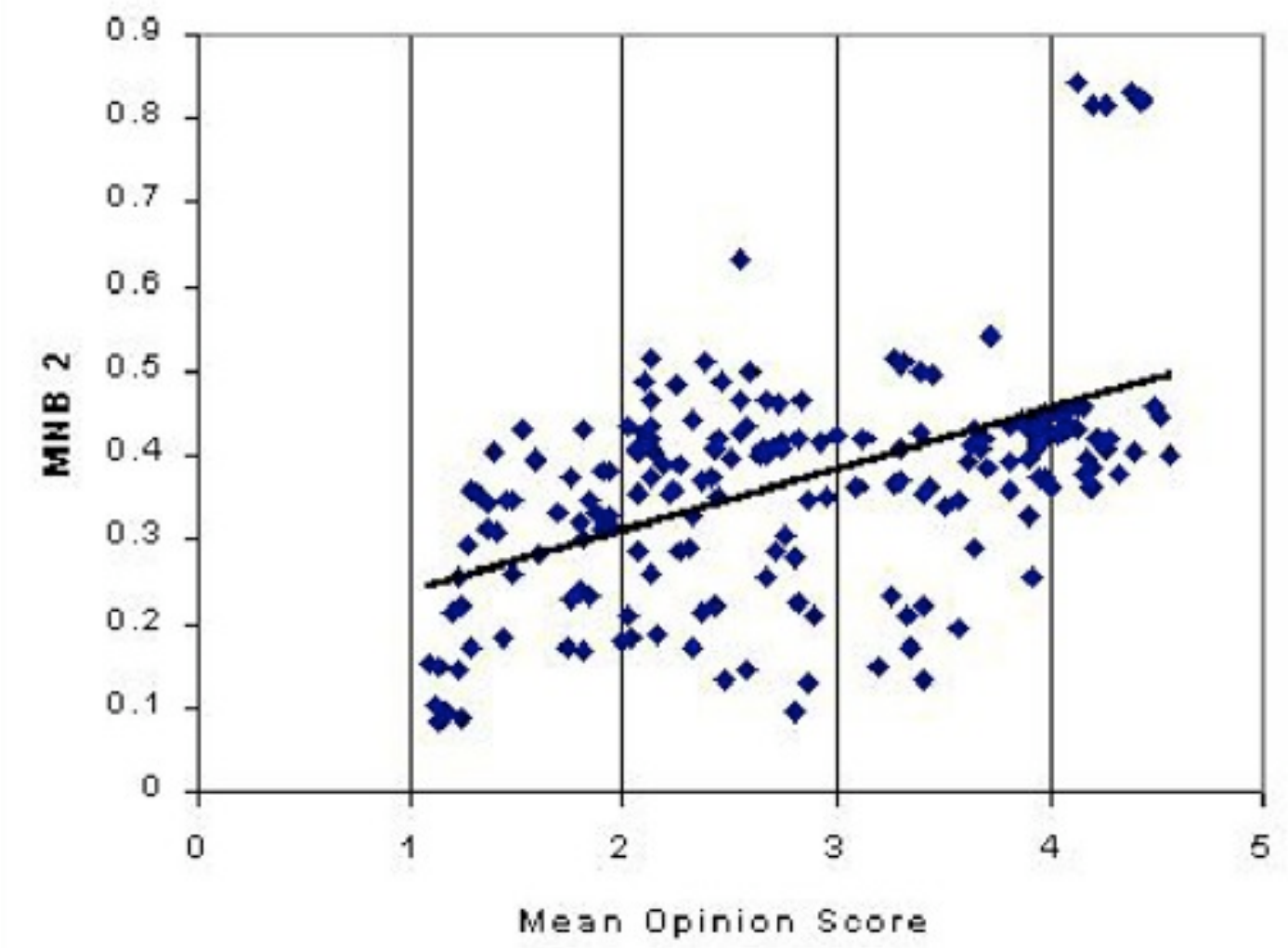
- Our problem: to quantify the perceived quality of a video or audio or multimedia stream transmitted over the Internet, automatically (and in real time if useful) and accurately.
- This task can be efficiently done with real human observers.
- Name of the field: *subjective testing*. Standards exist.
- Subjective testing is, by definition, not real time, nor automatic.

Objective tests

- Objective testing: basically, from tools to compare the original and the encoded stream (both at the source).
- It takes the form $d(\sigma, \sigma')$ where σ is the original sequence, σ' the encoded one, and $d()$ is some distance function.
- The original sequence is thus needed.
- A few exceptions exist (e.g. in voice streams, the E-model).
- Some tentative objective measures taking the form of a simple formula of one or two parameters have been published.
- So far, common and critical (negative) point: objective measures (anyway) correlate often badly with subjective testing.
- Some very recent work appears to get good results (good correlations with subjective tests). However,
  - the results concern mainly specific types of quality problems,
  - and the associated procedures are computationally heavy (for instance, impossible to embedded in a small mobile device).

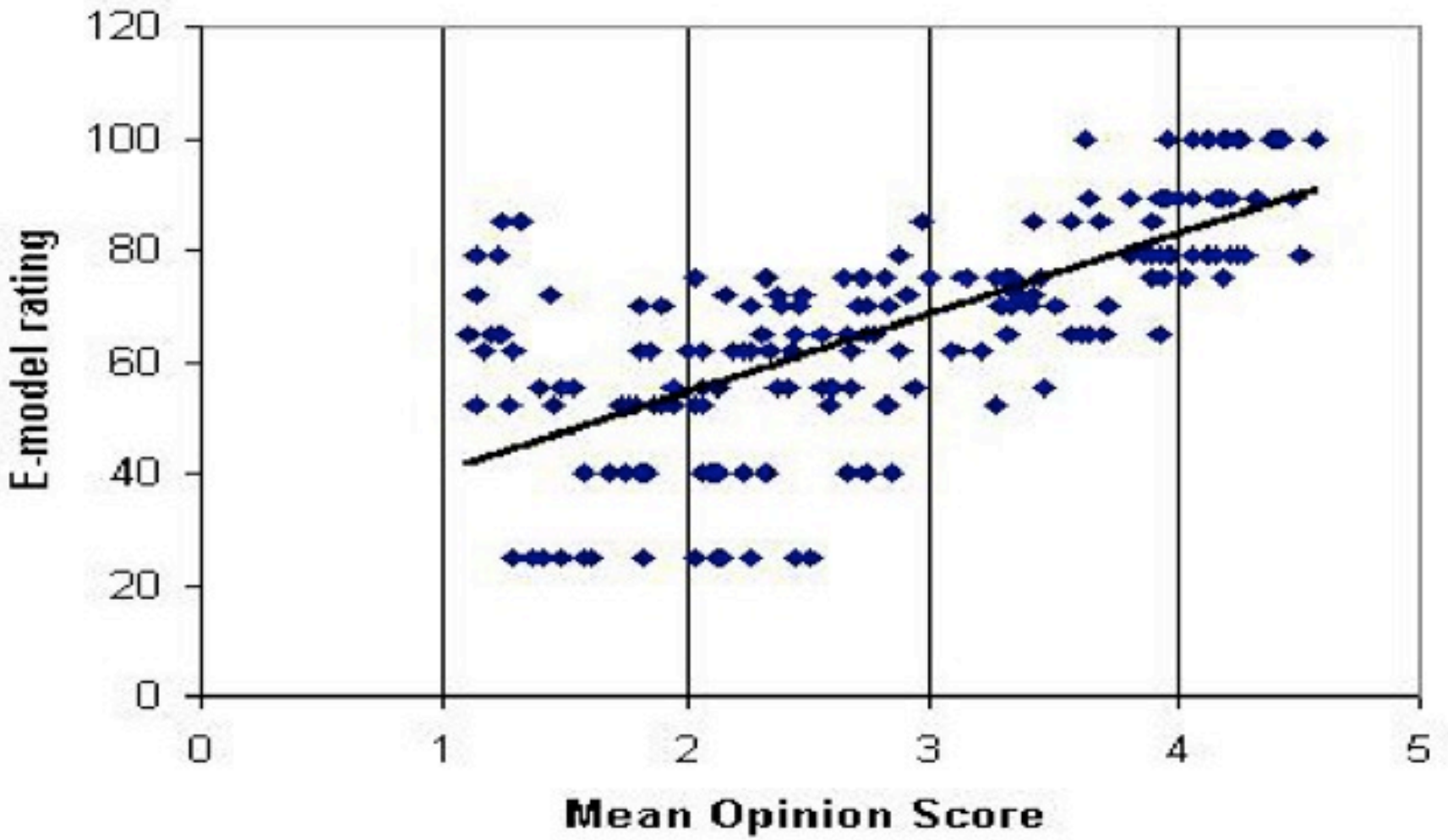
Example in audio

Bad correlation between an objective metric and subjective tests in audio.



Example in VoIP

Bad correlation between an objective metric and subjective tests in Voice over IP.



Pseudo-Subjective Quality Assessment (PSQA)

- PSQA aims to solve the problem of automatically providing an estimate of the perceived quality after transformation of the sequences by the network.
- It is based on a learning phase in which a specific tool captures the way humans react to the flows, from the quality point of view.
- The experimental work done so far with the tools shows accurate results.

PSQA step by step

- Choose a set of *a priori* relevant measurable factors.
  Example: our choice for video was
  - BR: source bit rate
  - FR: source frame rate
  - RL: redundancy level at the source
  - LR: packet loss probability in the network
  - MLBS: mean packet loss burst size in the network
- For each parameter, select a set of typical/important values.
  E.g., for LR: $\{0\%, 1\%, 2\%, 3\%, 5\%, 10\%\}$.
- This defines a state space for the vector of parameters.
  E.g. in our video case, around 3000 points.
  Call *configurations* these points.

PSQA step by step (cont'd)

- Select a sample set of size N (e.g., $N = 100$) points (this must be done with a merge of random sampling and covering criteria – like in low-discrepancy sequences).
- Select randomly N' points between this sample. E.g. $N' = 80$. Renumber them $1..N'$; the $N - N'$ remaining configs. are $N' + 1..N$.
- Build a testbed or a simulator allowing to control the chosen parameters.
- Start from a sequence σ (say, 10 sec. length) and send it N' times to the receiver according to the N' different configs.
  The result is a set $\{\sigma_1, \cdots, \sigma_{N'}\}$ of distorted copies of σ.

PSQA step by step (cont'd)

- Put K humans ($K \approx 20$) to evaluate these N' sequences according to a standardized subjective test.
  Let q_{ik} be the value given to sequence σ_i by observer k.
- Perform a statistical filtering of the $\{q_{ik}\}$ having as a goal to detect the bad observers, and eliminate the associated evaluations.
  Index again the remaining K' observers from 1 to K'.
- "The" value of σ_i is then $q_i = \frac{1}{K'} \sum_{k=1}^{K'} q_{ik}$.

PSQA step by step (cont'd)

- Train a RNN model having 5 inputs to obtain, for all $i = 1..N'$,

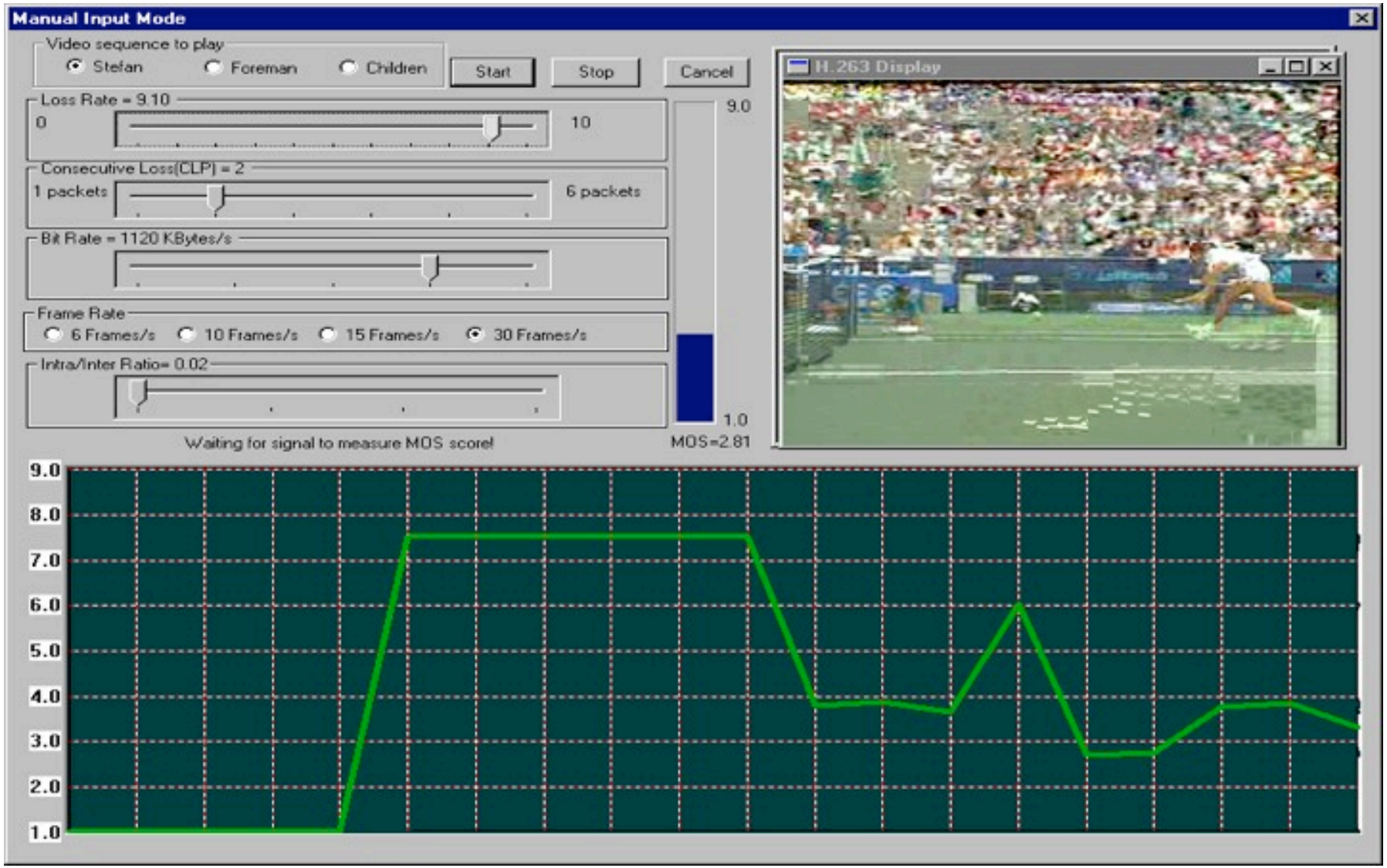
$$\nu(\mathrm{BR}_i, \cdots, \mathrm{MBLS}_i) \approx q_i.$$

- Validate the model if, for all $i = N' + 1..N$,

$$\nu(\mathrm{BR}_i, \cdots, \mathrm{MBLS}_i) \approx q_i.$$

- Improvement: perform the same process for several different sequences instead of just for only one.

(From a demo tool in video analysis)



Mathematically

- Our learning tool is a Random Neural Network (RNN). Extensive tests show that, for our use, it performs much better than other standard tools such as Artifical (Standard) Neural Networks (see our paper in ICANN'05).
- The PSQA function provided by our learning tool belongs to a specific class of rational functions.
- An integer-valued parameter H controls the degree of numerator and denominator. The higher H, the higher these degrees.
- There is basically an "optimal" value H_0, quite low in all the considered applications.

Some remarks

- A change in the network and/or in the applications considered can lead to the need of rebuilding everything. Our tools are related to a specific pair network/application (or to specific families of pairs).
- So far we didn't take into account environment parameters, e.g. noise levels.
- Let us underline again that we focus on the network side (its control, the services running on it, ...). That is why so far we haven't look at the signals directly:
  - analysis of blockiness, bluriness, fluidity...,
  - but also things such as the fact that suddenly there is no audio signal, the fact that after zapping there is a delay associated with the specific transmission system, for re-srating to play, the fact that black screens may appear, ...

Outline

As a stochastic dynamical system

- A Random Neuron is an input-output system, with two input ports, called *positive* and *negative*, a single output port and a positive integer-valued internal state, its *potential*.
- Each input port receives a random train of *pulses*.
- At time 0, there is some initial potential n_0.
- Each time a pulse arrives at the positive port, the potential increases by 1; each time a pulse arrives at the negative port, the potential decreases by 1 (except if it was 0, in which case it remains 0).
- The neuron is *active* iff its potential is > 0.
- If the neuron is active at t, it *fires* after an exponentially distributed delay; firing means sending out a pulse and decreasing its (necessarily > 0) potential by 1.

As a queue

- A Random Neuron can be also seen as a queue (in such a case we say a G-queue).
- Basically, only the vocabulary changes.
- Instead of positive (resp. negative) signals, we use positive (resp. negative) customers.
- Instead of "the potential of the neuron" we use "the number of customers in the queue" (or its backlog).
- Instead of "the neuron fires at t", we use "there is a service end at t".
- Observe that these queues only store and process positive customers. Negative customers can not be observed. Only their effects can.

Transient behavior

- Assume the pulses arriving at the positive (resp. negative) port follow a Poisson process with rate λ^+ (resp. λ^-).
- Let $\mu > 0$ be the firing rate of the neuron.
- Let N_t be the random variable "potential at t", and denote $p_n(t) = \Pr(N_t = n)$.
- Then, the family $(p_n())$ of functions satisfies the following (infinite) linear differential equation system: for $n \geq 1$,

$$p_n'(t) = p_{n-1}(t)\lambda^+ + p_{n+1}(t)(\lambda^- + \mu) - p_n(t)(\lambda^+ + \lambda^- + \mu),$$

$$p_0'(t) = p_1(t)\mu - p_0(t)(\lambda^+ + \mu),$$

and $p_{n_0}(0) = 1$.

Transient behavior (cont'd)

- To get an insight about the complexity of this, the most compact known closed expression of this distribution is[1], in the simplest $n_0 = 0$ case:

$$p_n(t) = \left(\frac{p}{q}\right)^n \sum_{j=n}^{\infty} e^{-\psi t} \frac{(\psi t)^j}{j!} \sum_{k=0}^{\left\lfloor \frac{j-n}{2} \right\rfloor} \frac{j+1-2k}{j+1} \binom{j+1}{k} p^k q^{j-k}$$

  where $p = \lambda^+/(\mu + \lambda^-)$, $q = 1 - p$ and $\psi = \lambda^+ + \lambda^- + \mu$.

- The general case is written as a function of the case of $n_0 = 0$, using modified Bessel functions of the first kind.

[1] P. Leguesdron, J. Pellaumail, G. Rubino, B. Sericola, *"Transient analysis of the M/M/1 queue"*, Advances in Applied Probability 25, pages 702–713, 1993.

Steady-state

- The stochastic process (N_t) is stable iff $\lambda^+ < \lambda^- + \mu$. In that case, for all n_0,
  - $\lim_{t\to\infty} p_n(t) = (1-\rho)\rho^n$, where $\rho = \lambda^+/(\lambda^- + \mu) < 1$;
  - in particular, the probability in steady-state that the neuron is active, its *activity rate*, is ρ.
- The fact that the activity rate of the neuron is the ratio $\lambda^+/(\lambda^+ + \mu)$ also holds when the arrivals are general point processes and the firing time is not exponentially distributed; it is a consequence of a general mean flow conservation theorem[2]. In this case the parameters λ^+, λ^- and μ are mean intensities.

[2] G. Rubino, *"Quantifying the Quality of Audio and Video Transmissions over the Internet: the PSQA Approach"*,
in *Design and Operations of Communication Networks: A Review of Wired and Wireless Modelling and Management Challenges*, Imperial College Press, Ed.: J. Barria, 2005.

Outline

Definition

- Let as use the queuing vocabulary now.
- A G-network (or a Random Neural Network) is an open (Jackson-like) network of queues with "positive" (standard) and "negative" (anti-matter-like) customers (so, a 2-class queuing network).
- These tools have been invented by Erol Gelenbe in the early 90s.
- Arrival flows at queue i from outside are Poisson. We denote by λ_i^+ (resp. by λ_i^-) the arrival rate for positive (resp. negative) customers.
- Queues only store and process positive customers.
- When a negative unit arrives at a queue, it destroys itself and if there are positive customers in the queue, one of them (say, the last in the queue assuming they are FIFO systems) is also destroyed.
- Again, negative customers can not be observed. Only their effects can.

Definition (cont'd)

- Services at the queues are exponentially distributed; μ_i is the service rate at queue i.
- After being served at queue i, a (necessarily positive) customer is sent outside with probability d_i, or it is sent to queue j as a positive one (resp. as a negative one) with (routing) probability r_{ij}^+ (resp. r_{ij}^-).
- Transfers between queues are instantaneous.
- If the network has N nodes,

$$d_i + \sum_{j=1}^{N} r_{ij}^+ + \sum_{j=1}^{N} r_{ij}^- = 1.$$

- Assume the network is in equilibrium and that ρ_i is the load at i (ρ_i is the probability that queue i is not empty).

Traffic equations

- Denote T_i^+ (resp. T_i^-) the mean throughput of positive (resp. negative) customers at i. We have (the traffic equations)

$$T_i^+ = \lambda_i^+ + \sum_{j=1}^{N} \rho_j \mu_j r_{ji}^+,$$

$$T_i^- = \lambda_i^- + \sum_{j=1}^{N} \rho_j \mu_j r_{ji}^- .$$

Product-form

- Gelenbe then proved the following theorem: if the (non-linear) system composed of the traffic ($2N$) equations plus the N equations

$$\rho_i = \frac{T_i^+}{\mu_i + T_i^-}$$

has a solution (in the throughputs and the loads) satisfying that for all node i, $\rho_i < 1$, and with the usual connectivity assumptions, then the solution is unique, the model is ergodic and it has a product-form solution.

- As a consequence of the previous theorem, the ρ_is solution to the previous system are the loads at the different nodes.

Product-form (cont'd)

- The product-form solution means that for any initial condition (on the numbers of customers at the different nodes at time 0), we have

$$\lim_{t \to \infty} \Pr(n_i \text{ units in } i \text{ at } t) = C \, \Pi_{i=1}^{N} \rho_i^{n_i},$$

where

$$C = \Pi_{i=1}^{N} (1 - \rho_i).$$

Outline

Approximating the quality function

- We look at the perceived quality as an (unknown) real function $f()$ of I variables.
- As our variables will be always bounded, we scale everything (the input variables and the measure of quality) on $[0..1]$.
- Our goal: to approximate function $f()$ using a set of known values, following a learning process.
- Our learning tool will be a feedforward RNN.

Feedforward RNN

- Our approximation tool will be a G-network having $I + H + 1$ nodes with a feedforward architecture (so, better called a Random Neural Network, RNN).
- Nodes $i = 1..I$ receive external positives customers from outside.
  The rate of the Poisson arrival process at node i is λ_i, assumed to be in $[0, 1]$.
  There are no negative arrivals from outside.

Feedforward RNN (cont'd)

- Node i sends its customers to nodes indexed by $h = I + 1..I + H$. The routing probabilities are r_{ih}^{+} and r_{ih}^{-}.
- Nodes $h = I + 1..I + H$ send their customers to node $o = I + H + 1$, and the associated routing probabilities are r_{ho}^{+} and r_{ho}^{-}.
- Node o sends its customers to outside.
- Here, the non-linear system allows a simple closed-form solution.

Feedforward RNN (cont'd)

- For inputs nodes $i \in \mathcal{I}$, $\rho_i = \lambda_i / \mu_i$.
- For hidden nodes h, denoting $w^*_{ih} = \mu_i r^*_{ih}$,

$$\rho_h = \frac{\sum_{i \in \mathcal{I}} \rho_i w^+_{ih}}{\mu_h + \sum_{i \in \mathcal{I}} \rho_i w^-_{ih}}.$$

- For the (only) output node, denoting $w^*_{ho} = \mu_h r^*_{ho}$,

$$\rho_o = \frac{\sum_{h \in \mathcal{H}} \rho_h w^+_{ho}}{\mu_o + \sum_{h \in \mathcal{H}} \rho_h w^-_{ho}}.$$

Learning

- Recall that the goal is to approximate an unknown real function $f()$ having I variables. Recall that we will use scaled variables here:

$$f : [0..1]^I \rightarrow [0..1].$$

- Function $f()$ is known only through pairs $(\vec{a}_k, b_k)$, $k = 1..K$, where $f(\vec{a}_k) = b_k$.
- We look at the G-network as a real function $\nu()$ of I real variables. The I variables are the λ_i's and the output is $\nu(\vec{\lambda}) = \rho_o$, the load in equilibrium of node o, where $\vec{\lambda} = (\lambda_1, \cdots, \lambda_I)$.

Learning (cont'd)

- The service rates μ_i are fixed, and the routing probabilities are seen as parameters (through the notation change $w_{ij}^* = \mu_i r_{ij}^*$).
  Thus, we write $\rho_o = \nu(\vec{\lambda}; \vec{w})$ $(\vec{w} = \{w_{ij}^*\})$.
- The μ_i's can be set to assure stability, or to leave the unstable case as a possibility.
- The goal is to find parameters $\vec{w}$ such that the G-network behaves as $f()$. This process has two phases: the learning phase and the validation phase.

Learning (cont'd)

- Having the K data pairs in random order, they are decomposed into two sets: $\{(\vec{a}_k, b_k),\ k = 1..K'\}$ and $\{(\vec{a}_k, b_k),\ k = K'+1..K\}$.
- **Learning** $\equiv$ solving the optimization problem

$$\text{find} \qquad \vec{\pi} = \text{argmin}_{\vec{w} \geq \vec{0}} \sum_{k=1}^{K'} (b_k - \nu(\vec{a}_k; \vec{w}))^2.$$

Learning (cont'd)

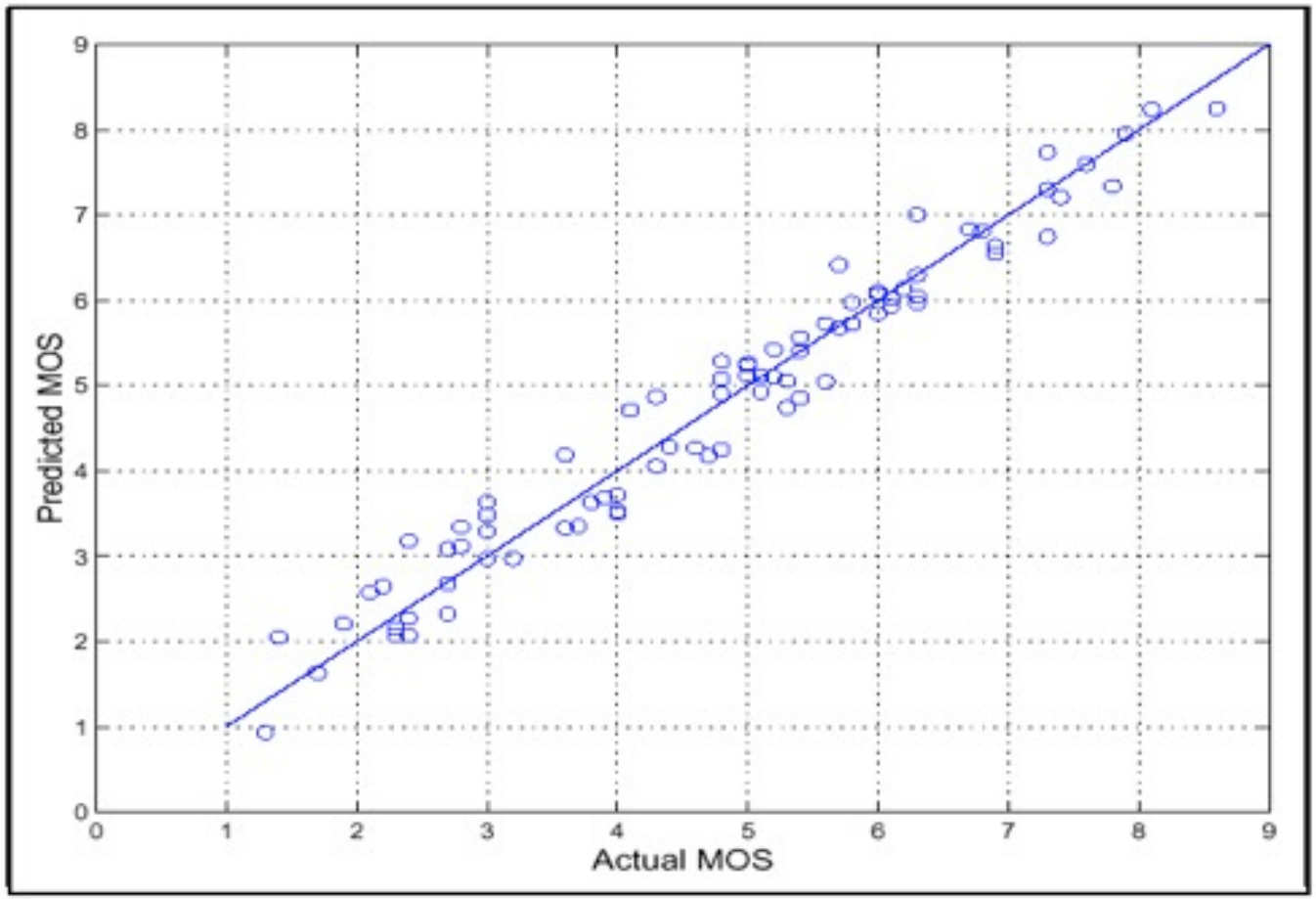
- If
  - the $\vec{a}_k$'s, for $k = 1..K'$ "reasonably" cover the possible input values of $f()$ (say, the domain $\mathcal{D} = [0,1]^l$),
  - and if $f()$ has some properties (bounded derivatives, ...)

  we expect the RNN $\nu(.;\vec{\pi})$ accurately approximating $f()$ over the whole $\mathcal{D}$.
- **Validation** $\equiv$ evaluating

$$\text{Err} = \sum_{k=K'+1}^{K} (b_k - \nu(\vec{a}_k;\vec{\pi}))^2$$

and "accepting" $\nu(.;\pi)$ if $\text{Err} < \varepsilon$.

Training results for PSQA (video example)

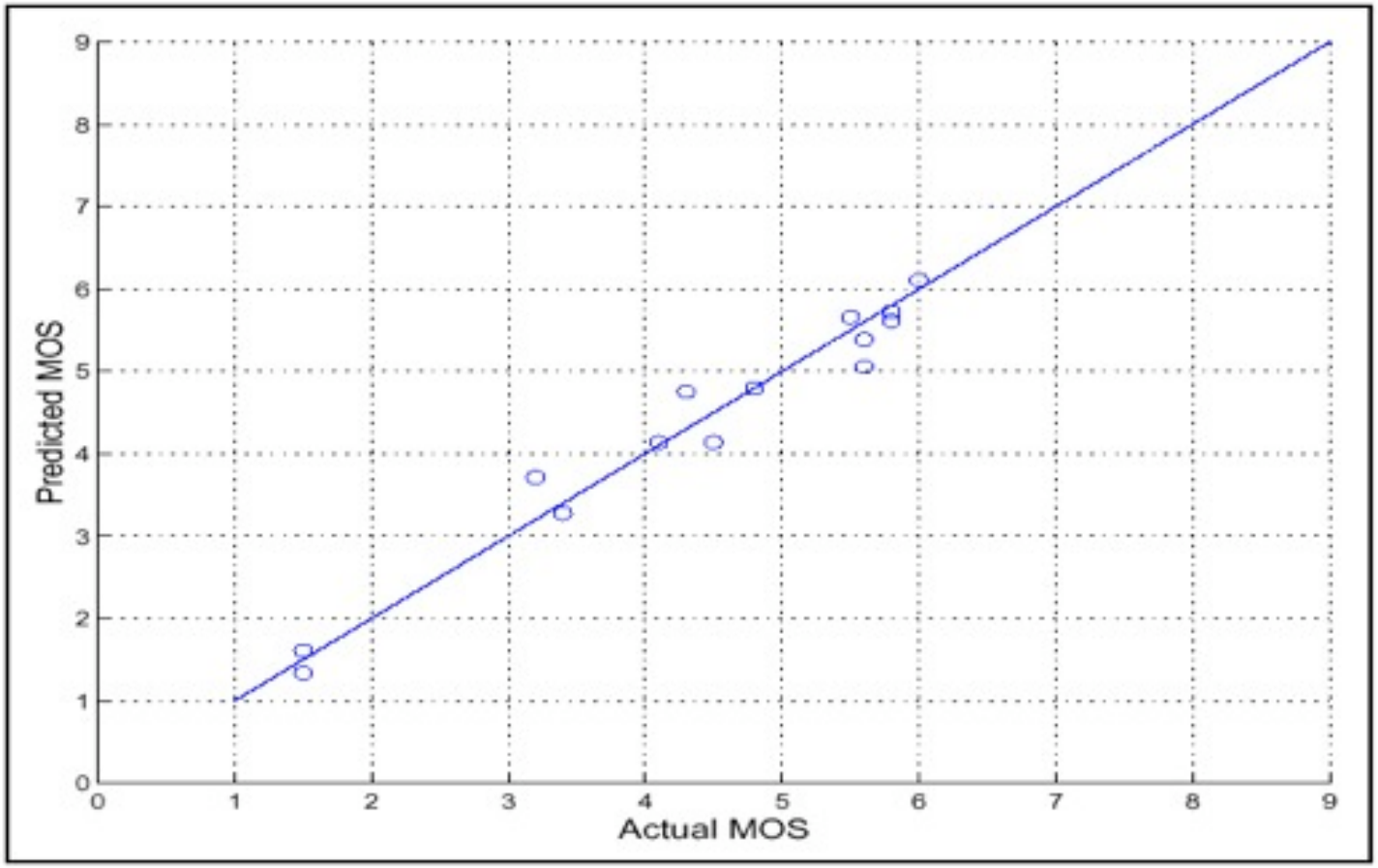


(a) Training DB

(Results are not surprising.)

Validation results for PSQA (video example)

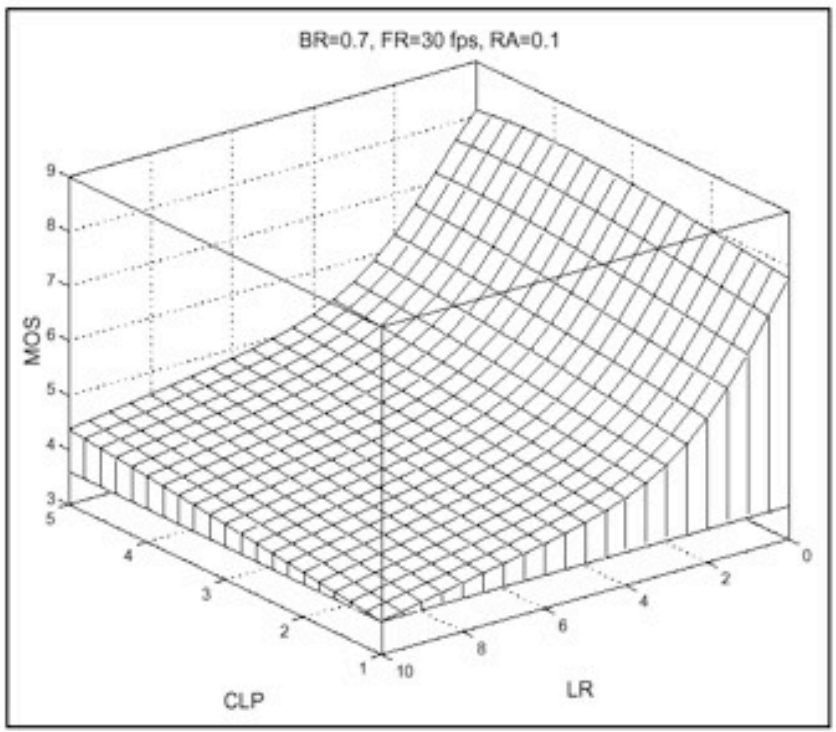
Very good accuracy of the tool:



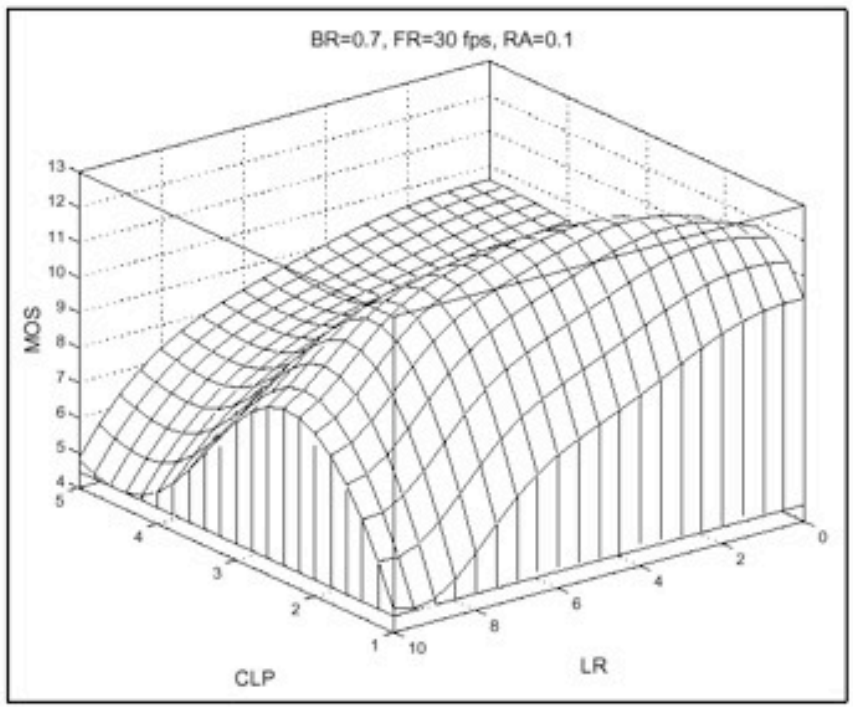
(b) Testing DB

Comparing RNN with ANN

Under exactly the same (reasonable) conditions and with the same data (an example, there are many other ones):



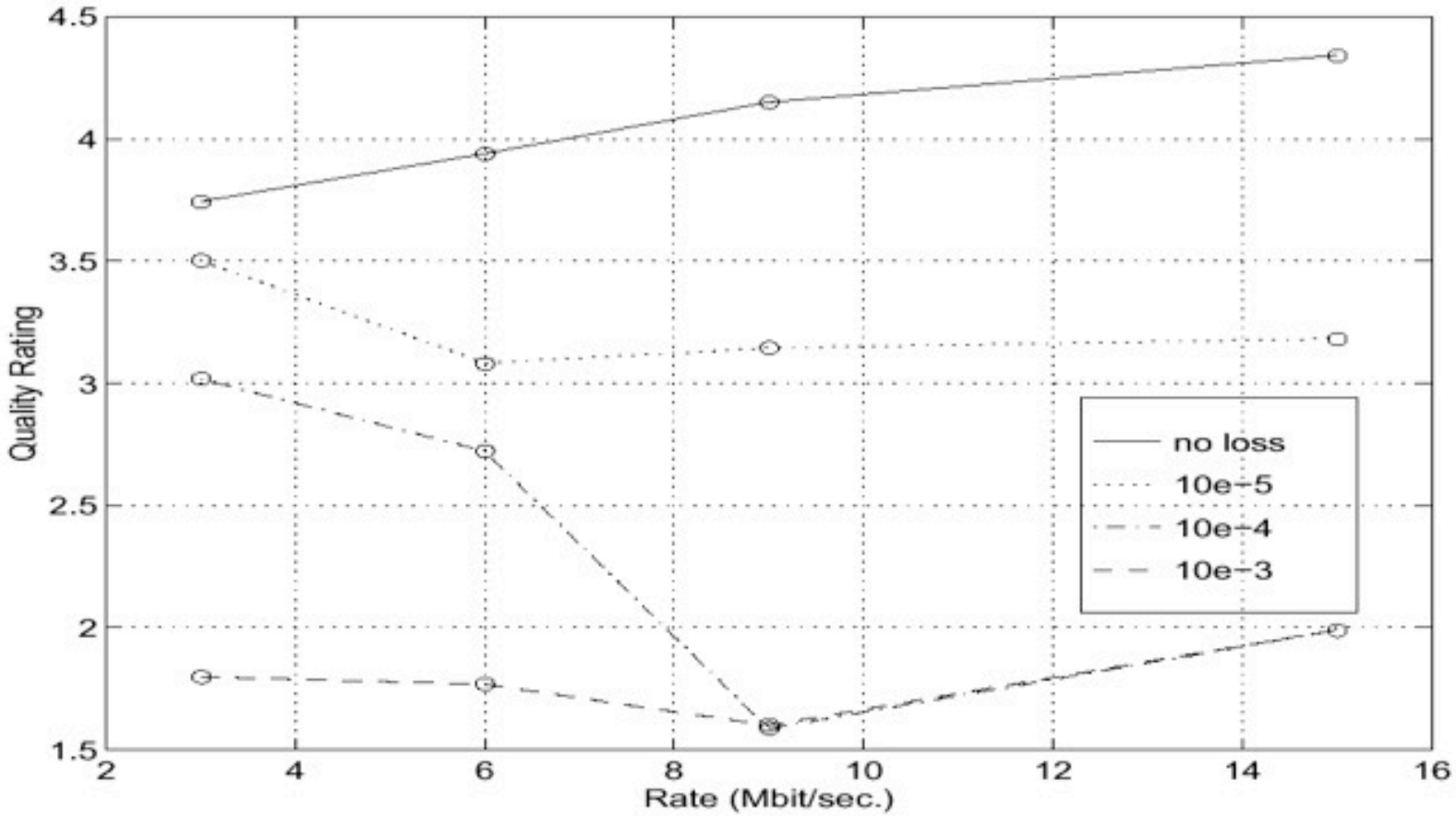
(a) Correctly trainined



(b) Example of an over-trained ANN

Another type of observation for comparison purposes

An example of bad behavior of an objective measure (PSQA behaves exactly as expected, "by construction"): MPQM (well-known metric for voice) when applying losses, as a function of bit rate.



Outline

Main applications

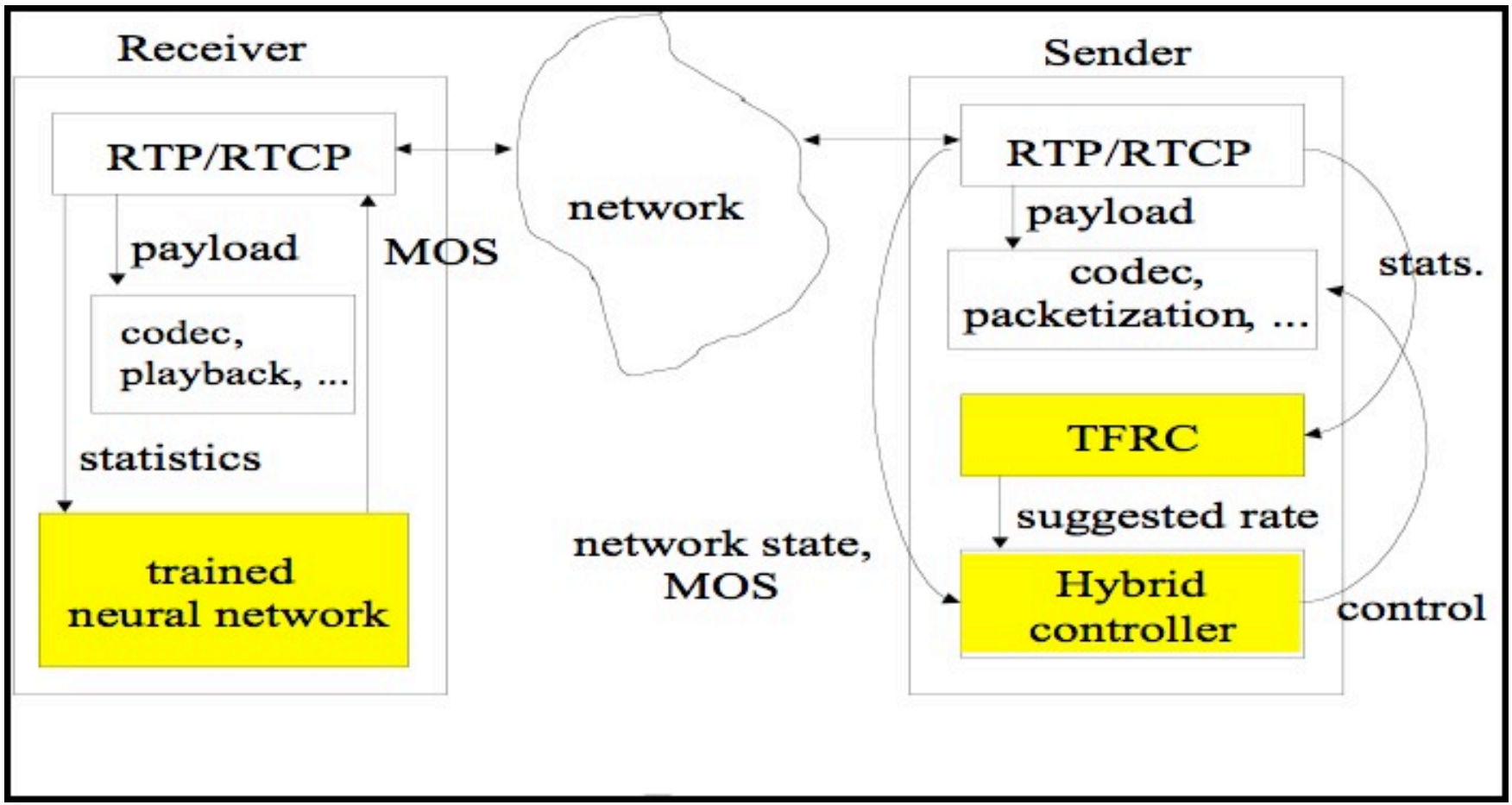
- network analysis
- network control
- network design
- network modeling

About network control

- Put a PSQA module at some strategic points in the network, or some specific terminals, capable of sending reports to some decision center(s).
- Use the instantaneous quality level observed at those points to take control decisions on the network. These can be, for instance,
  - changing the parameters of the transmission (codec, ...)
  - reconfiguring a part of the network (for instance, in a P2P context),
  - changing the routing procedures,
  - ...

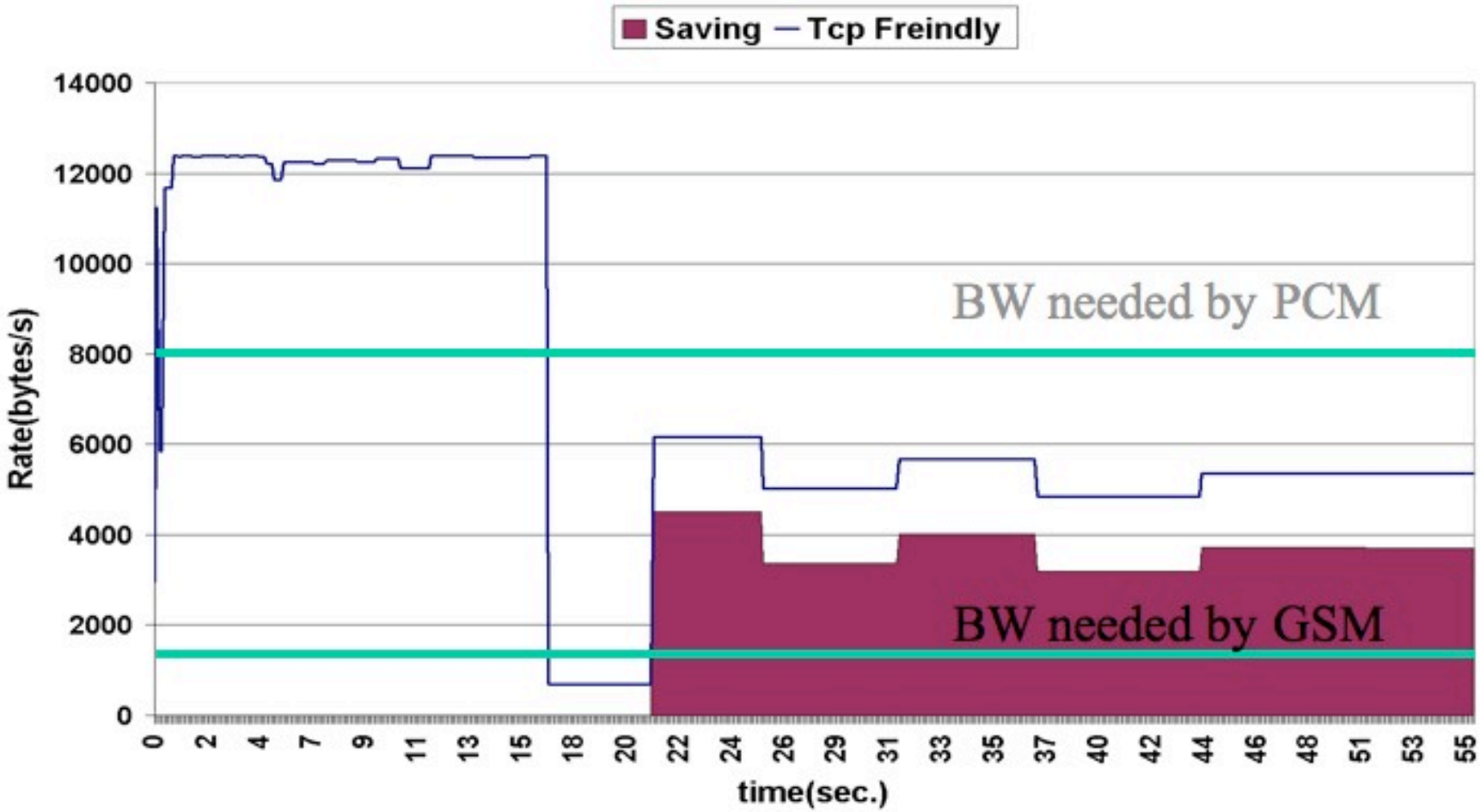
Controlling voice transport: a toy example

The general setting:



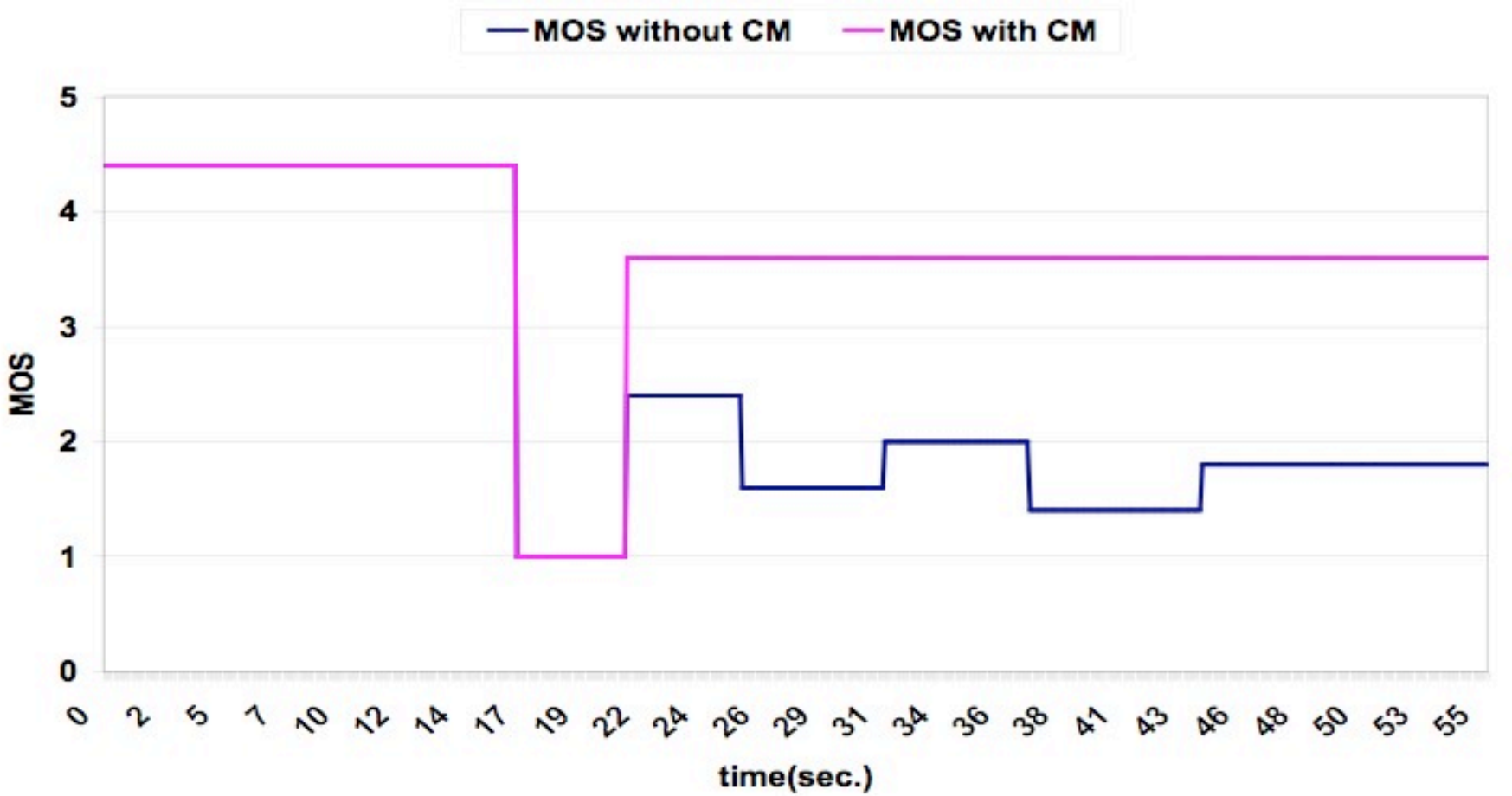
Controlling voice transport: a toy example (cont'd)

Bandwidth evolution:



Controlling voice transport: a toy example (cont'd)

Using PSQA:



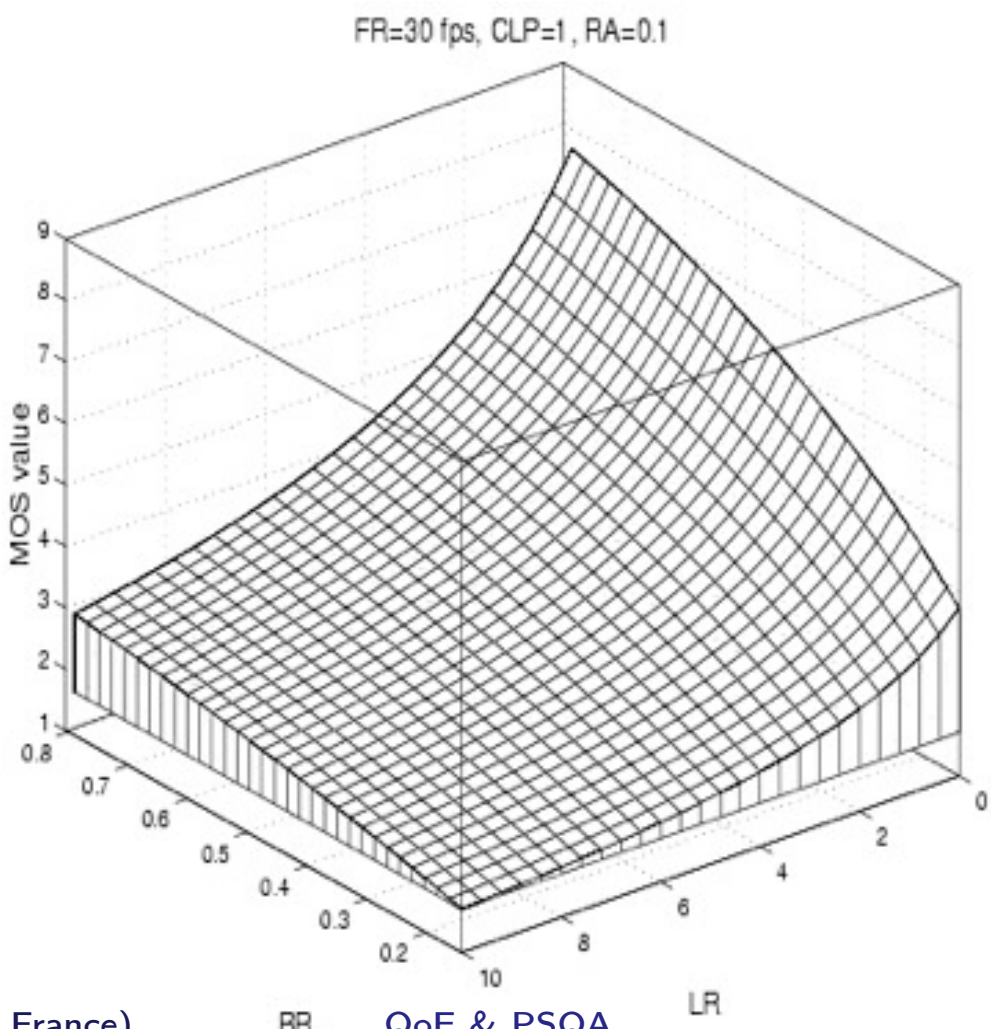
A project involving network design and control

- a large project (1 PhD, 3 Msc)
- design of a P2P-based live-video distribution system
- still an on-going project
- complete QoE-oriented design for optimizing QoE, the "ultimate target"
- PSQA also used to control the network (optimizing the network architecture)

Outline

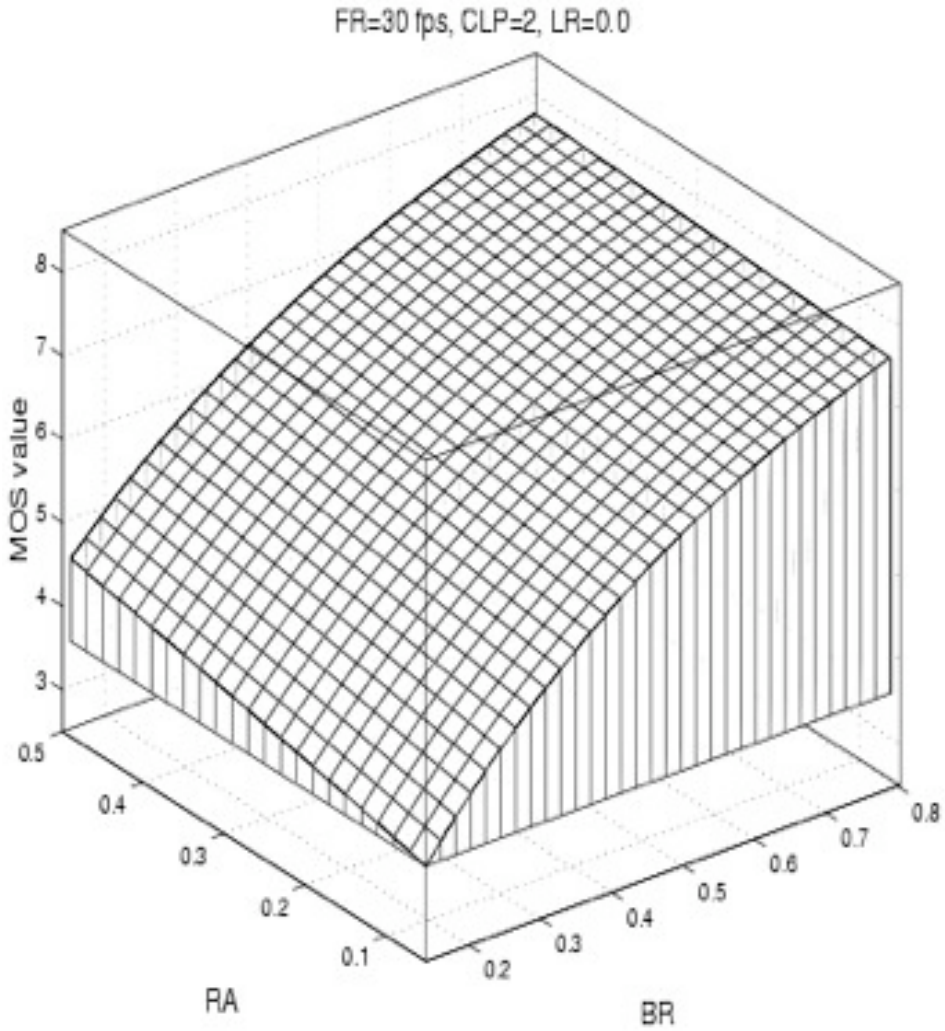
Network analysis: using PSQA

Once the PSQA module built, we can exploit it for understanding how the different parameters combine their effects on quality.
Example (video case):



Network analysis: using PSQA (cont'd)

Another example:



More recent work

How does quality react...

- To an increase in loss rate?
- To variations of the motion in the video?
- To the addition of redundancy by the sender?
- To an increase of buffering capacity in the receiver?
- To a combination of the points above!...?

We have being using PSQA to answer these questions and others, in two different contexts.

Two PSQA functions

A simple function (used in some theoretical studies)

- MPEG-2 encoding
- 100 video sequences
- first study made with "frame level" parameters
- only distribution-oriented parameters considered

A more complex function (used in our current prototypse)

- MPEG-4 (Xvid) encoding
- 204 video sequences
- at "frame level", discriminating frame type: I,P and B
- source–oriented and distribution-oriented parameters

Our MPEG-2 PSQA simple function: loss rate distribution

Two parameters

two network-oriented input variables (that is, we fixed the characteristics of the stream, such as bandwidth, encoding,. . .):

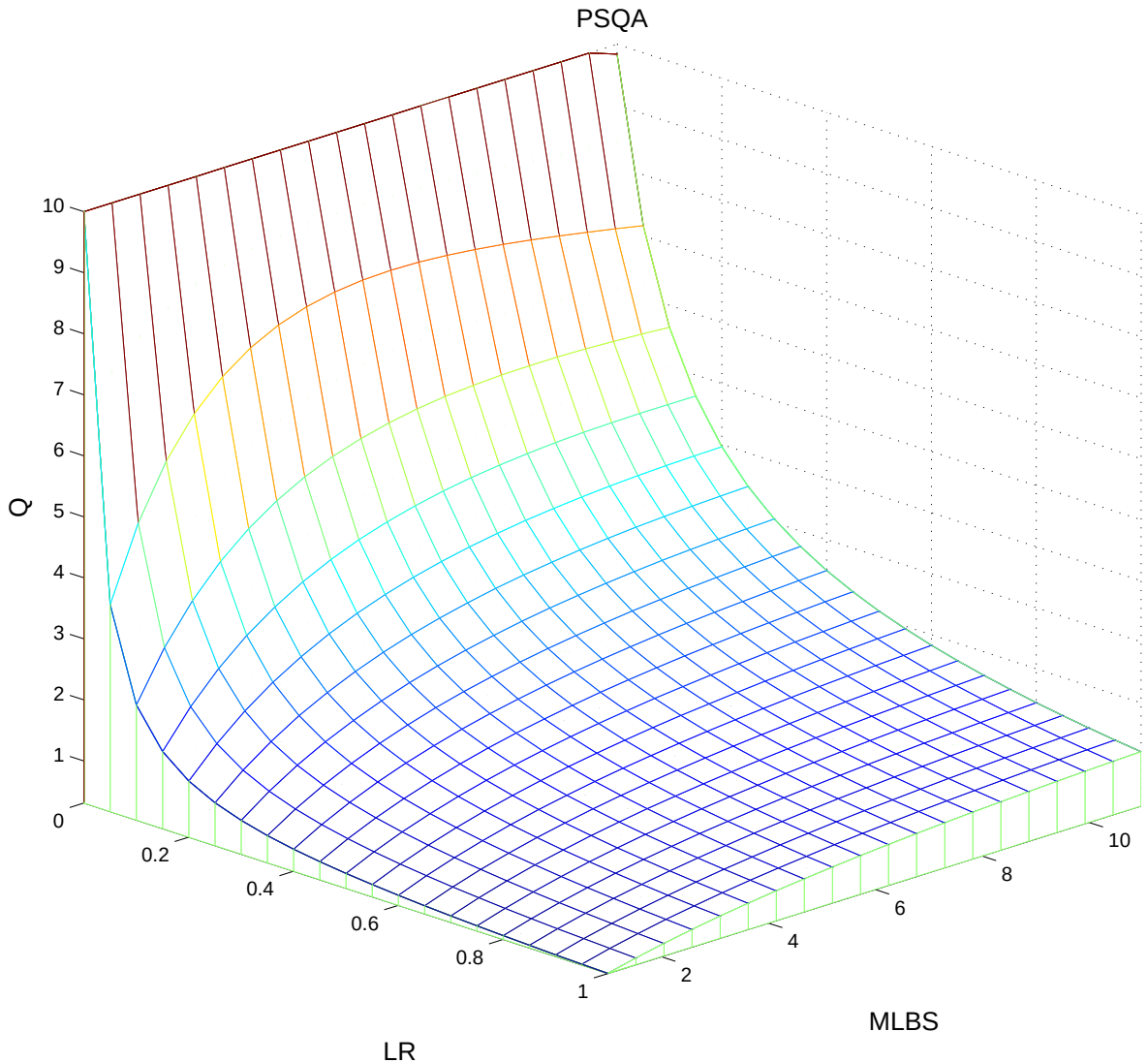
- the frame loss rate, denoted by LR
- the mean size of the bursts of frame losses, denoted by $MLBS$

We consider. . .

- LR from 0.0 to 0.2 (quality is too bad after 20% of losses)
- $MLBS$ from 1 to 10 frames

Our MPEG-2 PSQA simple function: loss rate distribution

[left,top]



- observe the monotonicity of Q with LR and $MLBS$
- in particular, the worst quality corresponds to the value $MLBS = 1$
- observe the less sensitivity of Q w.r.t. $MLBS$

Our PSQA *complex function*: frame types

Five parameters:

- network–oriented parameters: frame losses per type LR_I, LR_P, LR_B
- source–oriented parameters: the video motion (different metrics tested)
  GOP size and *frames P information ratio*

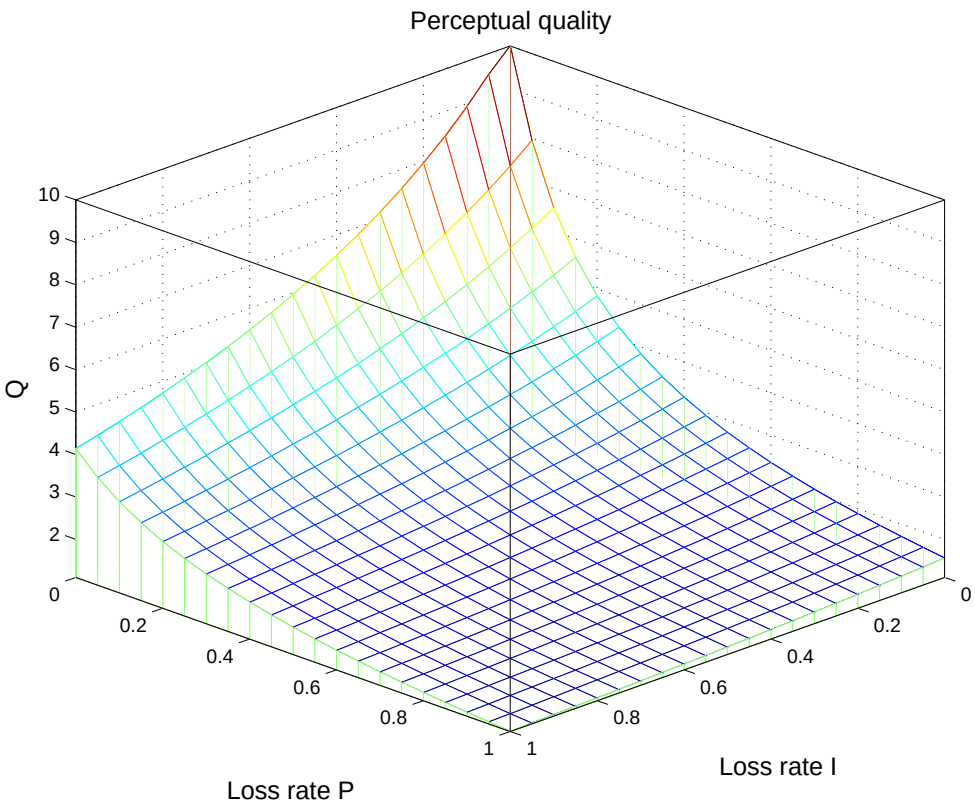
We consider

- LR_I from 0.0 to 1.0
- LR_P and LR_B from 0.0 to 0.25
- *GOP size* from 25 to 350 frames
- *frames P information ratio* from 0.05 to 0.9

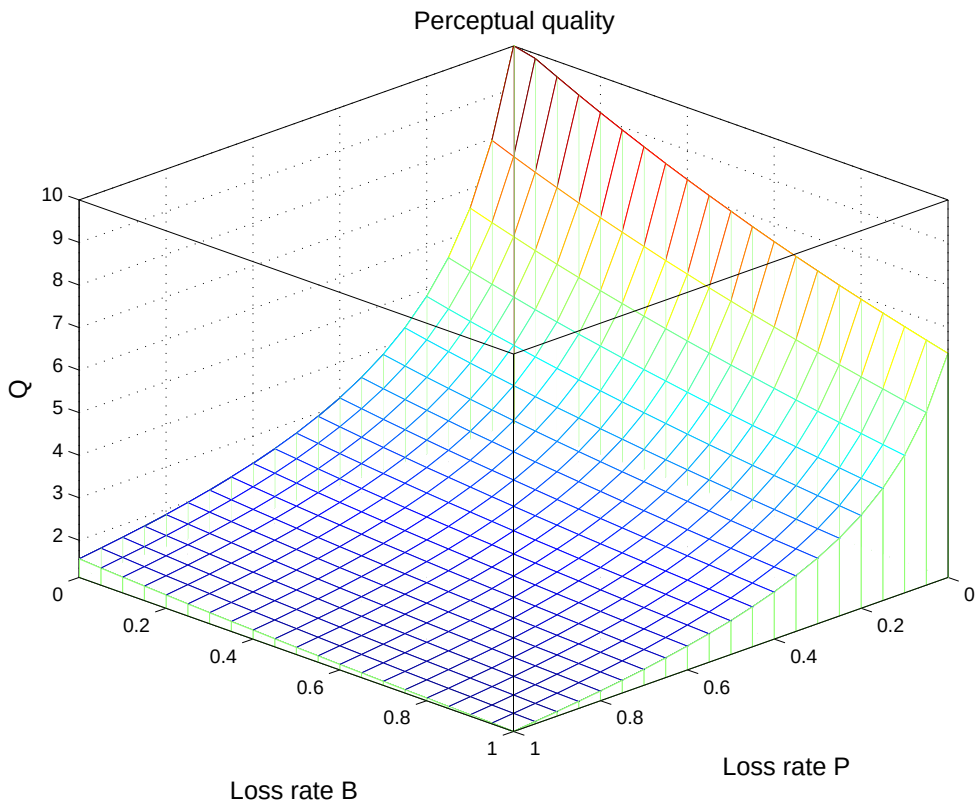
Our PSQA *complex function*: frame types

$$LR_I, LR_P, LR_B, \text{motion} \mapsto \text{quality}$$

[left,botton]



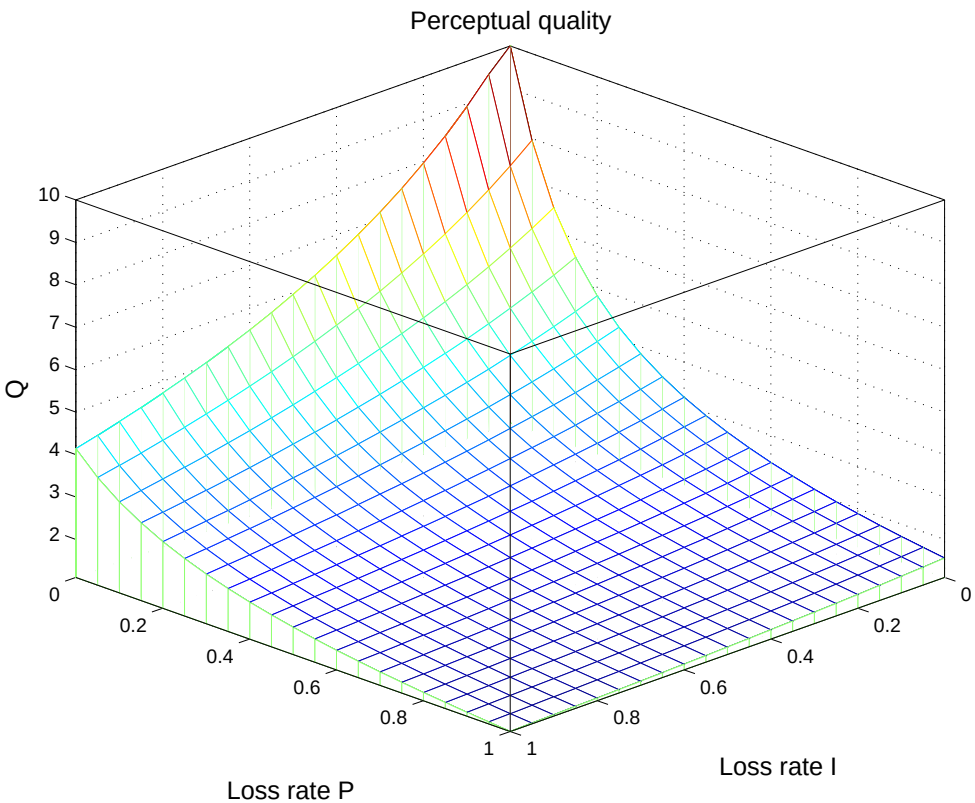
[rigth,top]



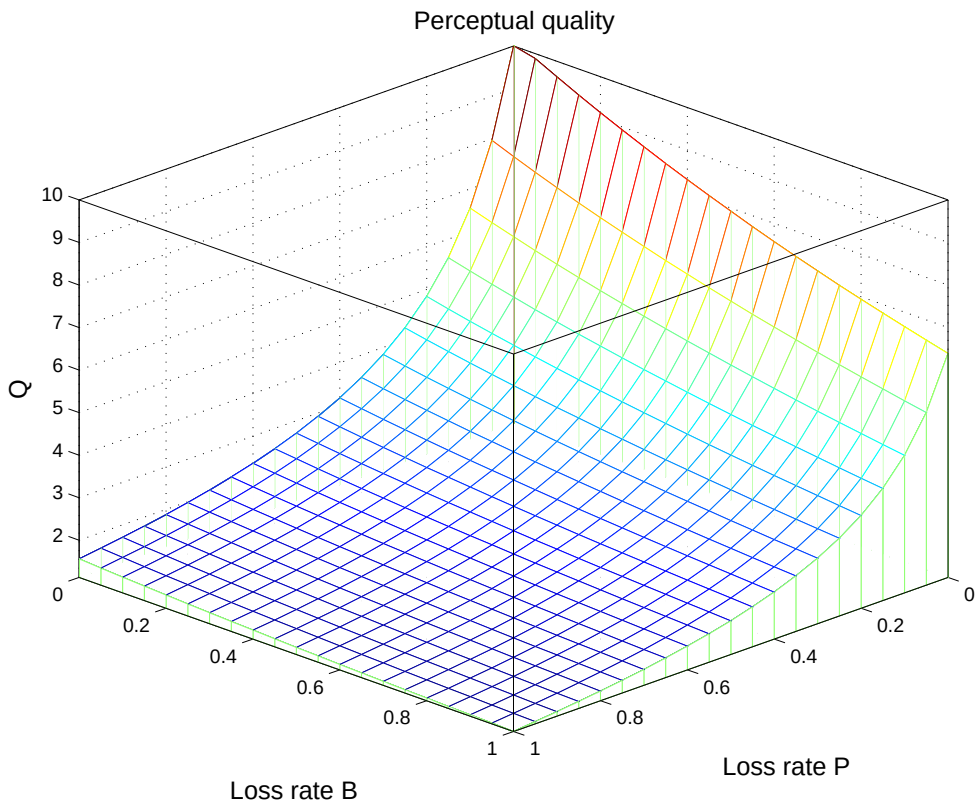
Our PSQA *complex function*: frame types

$$LR_I, LR_P, LR_B, \text{motion} \mapsto \text{quality}$$

[left,botton]



[rigth,top]



- observe the monotonicity of Q w.r.t. LR's
- quality degrades quickly with LR_I and LR_P
- the impact of LR_P is a bit higher than for LR_I
- quality degrades slowly with LR_B

Outline

Standard modeling approach

- Goal: end-to-end performance evaluation of some stream transporting system.
- Assume the network (or its bottleneck) is represented by some classical stochastic model.
  For instance, assume for illustration purposes, the following unrealistic but simple assumptions:
  - the video flow arrives as a Poisson process wth rate FR fps to the queue,
  - which has a transmission rate of c bps;
  - frames have an average size of B bits;
  - the storage capacity is HB bits.

Standard modeling approach (cont'd)

- Standard approach: since quality strongly depends on the loss process, focus on the loss probability.
- Looking at the model at the packet level, this is a $M/M/1/H$ queue.
- Result: the loss probability is

$$\mathrm{LR} = \frac{1-\rho}{1-\rho^{H+1}}\rho^{H},$$

with $\rho = \frac{\mathrm{BR}}{c/B} = \mathrm{BR}/c$, assumed here to be $\neq 1$.

Using PSQA: example in video analysis

Proposed approach:

- In the $M/M/1/H$ model, a simple Markovian analysis gives: $\mathrm{MLBS} = 1 + \rho$.
- The quality as perceived by the user, as a function of the data BR, FR, RL, c and H is

$$\nu\left(\mathrm{BR}, \mathrm{FR}, \mathrm{RL}, \frac{1-\frac{\mathrm{BR}}{c}}{1-\left(\frac{\mathrm{BR}}{c}\right)^{H+1}}\left(\frac{\mathrm{BR}}{c}\right)^{H}, 1+\frac{\mathrm{BR}}{c}\right).$$

Using PSQA: example in VoIP

Similarly (the goal here is to use the same queueing model) consider the bottleneck of a VoIP connexion represented by a $M/M/1/W$ queue.

PSQA variables:

- source bit rate (in Kbps),
- length of the packets (in msec),
- FEC (Forward Error Correction) offset,
- network loss rate,
- network mean loss burst size.

Using PSQA: example in VoIP (cont'd)

For instance,

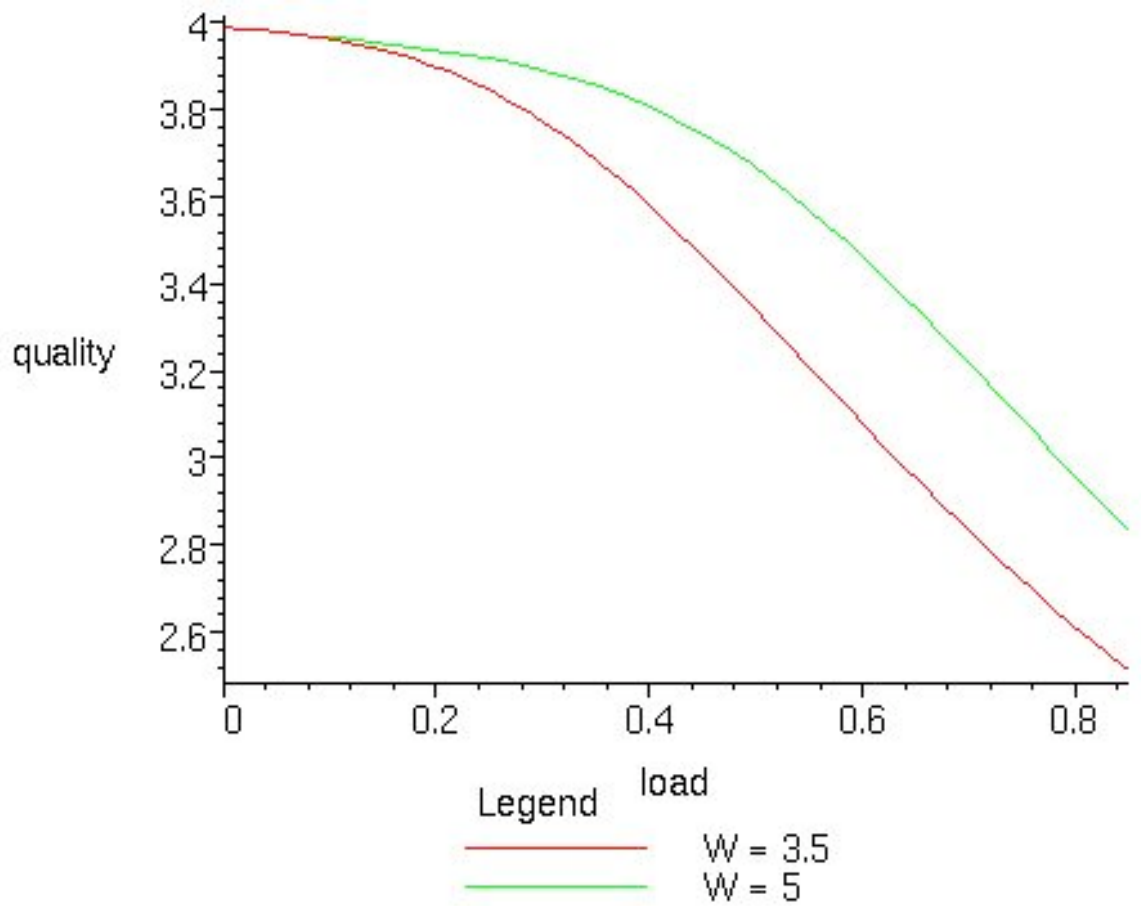
- using the PCM codec,
- FEC with an offset of 1
- and a packetization interval equal to 20 ms,

leaving only free the window size W and the load ρ, the perceived quality has the following form:

$$Q = \frac{\alpha + \beta\rho + \gamma\rho^W + (\gamma + \alpha)\rho^{W+1} + \beta\rho^{W+2}}{\alpha' + \beta'\rho + \gamma'\rho^W + (\gamma' + \alpha')\rho^{W+1} + \beta'\rho^{W+2}},$$

where $\alpha = 0.1326$, $\beta = 0.0201$, $\gamma = 0.0674$, $\alpha' = 0.1659$, $\beta' = 0.0399$, $\gamma' = 0.9326$.

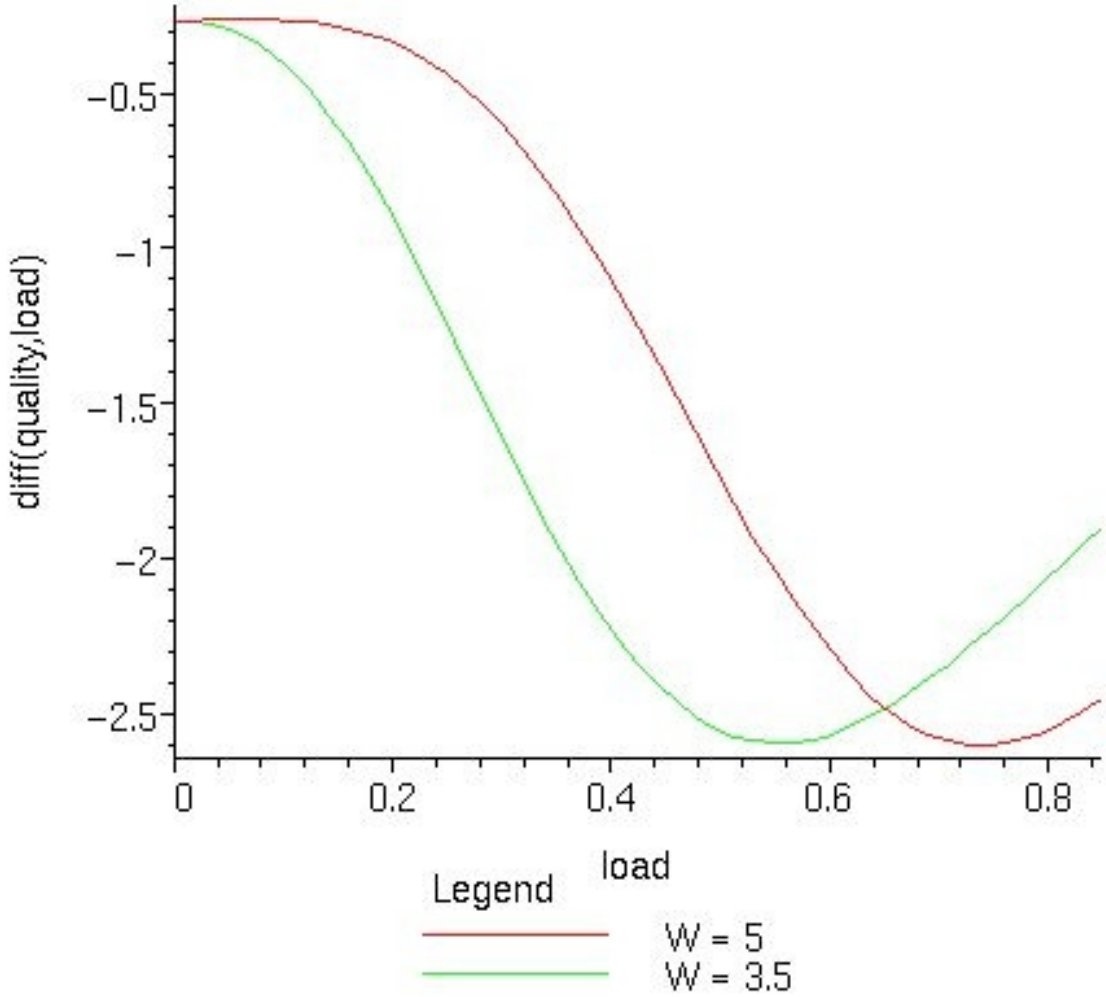
Using PSQA: example in VoIP (cont'd)



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

Using PSQA: example in VoIP (cont'd)

Having analytical expressions allows for other mathematical treatments. For instance, here is an example of sensitivity analysis (quality with respect to load):



Utility functions

- In many analysis, we must use utility functions representing the gain the user obtains from the network, as a function of some measured resource.
  E.g., $U = U(c)$.
  $U()$ is assumed to have some mathematical properties (e.g. concavity, derivatives exist at any order, ...)
- Using the previous approach and the fact that $\nu()$ (that is, the RNN model) has nice mathematical properties ($\nu()$ is a rational function), we can work with specific and meaningful utility functions.

Utility functions (cont'd)

- In the previous example, assume we use a very simple RNN with only $I + 1$ nodes (no "hidden layer").
- In this case, $\nu()$ is the ratio between two polynomials in the selected 5 variables having both degree one.
- If we need to look at the quality as a function of the bandwidth, we can fix BR, FR, RL, and W, and after some algebra, we obtain

$$Q = \frac{(\alpha + \beta c + \gamma c^2 - \delta c^{W+1} - \phi c^{W+2})(\psi - c^{W+1})}{(\alpha' + \beta' c + \gamma' c^2 - \delta' c^{W+1} - \phi' c^{W+2})(\psi' - c^{W+1})}$$

where α, β, ..., ϕ' are positive constants.

Utility functions (cont'd)

- Since we use normalized variables (everything is scaled to be in $[0, 1]$, so $c < 1$), we can use

$$Q \approx \frac{\psi}{\psi'} \frac{\alpha + \beta c + \gamma c^2}{\alpha' + \beta' c + \gamma' c^2},$$

or, simplifying,

$$Q \approx K \frac{1 + Ac + Bc^2}{1 + A'c + B'c^2},$$

etc.

Outline

Standard modeling approach

- Standard situation: evaluation of some dependability metric associated with a service (or more generally, with some specific aspect of the network).
- Examples of typical metrics: having defined what's an operating system is (user oriented definition),
  - reliability at $t = R(t) = \Pr(\text{the system works from 0 to } t)$,
  - MTTF = Mean Time To Failure = $\mathrm{E}(\text{system's life-time}) = \int_0^\infty R(s)\, ds$,
  - Point Availability at $t = PAV(t) = \Pr(\text{the system works at time } t)$,
  - Interval Availability on $[0, t]$ = fraction of $[0, t]$ during which system works,

$$IAV(t) = \frac{1}{t} \int_0^t 1(\text{system works at } s)\, ds,$$

  - Mean Time To Repair, ...

- Assume again the network (or its bottleneck) is represented by a classical stochastic model, using the unrealistic assumptions
  - the video flow arrives as a Poisson process wth rate FR fps to the queue,
  - which has a transmission rate of c bps;
  - frames have an average size of B bits;
  - the storage capacity is HB bits.

- Assume that the failure of some network component or components makes that the communication is stopped without losses, until the component is back to operation.
- Assume we can represent this making the server of the queue representing the bottleneck of the communication fail.
- Assume exponential service and repair times with respective rates f and r.
- Looking at the model at the packet level, this is a $M/M/1/H$ with failures and repairs, a "two-level" Markov chain that can be easily analyzed.

- A typical dependability metric here is the asymptotic availability of the system,

$$PAV(\infty) = \text{Pr(server is ``up'')} = \frac{1}{1+\psi}, \quad \psi = \frac{f}{r}.$$

- A typical "performability" ("performance + dependability") metric could be a conditional asymptotic availability informally defined as the probability that the server is up *when I need it*, that is, when there is some work for it, that is,

$$\text{Pr(server up} \mid \text{system busy)}, = \frac{1}{1+\psi}\frac{\rho(1-\rho^H)}{1-\rho^{H+1}}.$$

Coupling with PSQA

- We had an expression of the perceived quality as a function of 5 parameters, two of them related to the network, the loss probability LR and the mean loss burst size MLBS.
- The direct analysis of the Markov chain allows to derive the loss probability LR:

$$\mathrm{LR} = \frac{1}{\psi}\left[\psi + \frac{\rho(1-\rho^{H})}{1-\rho^{H+1}}\right].$$

- The computation of the MLBS is much more involved and the result is complex; it is based on the analysis of sojourn times in Markov models and of Palm measures (say, looking at the Markov model at specific points in time). Denote $\mathrm{MLBS}(\rho,\psi)$ the obtained result.

- As a result, we have (an accurate approximation/representation of) the quality as perceived by the user, as a function of the data BR, FR, RL, c, H and ψ:

$$\nu\left(\mathrm{BR}, \mathrm{FR}, \mathrm{RL}, \frac{1}{\psi}\left[\psi + \frac{\rho(1-\rho^{H})}{1-\rho^{H+1}}\right], \mathrm{MLBS}(\rho,\psi)\right).$$

First, some papers

- For details about the analysis of the impact of different factors on quality:
  - for video, see (our original paper) **A Study of Real-Time Packet Video Quality Using Random Neural Networks**, S. Mohamed and G. Rubino, *IEEE Transactions on Circuits and Systems for Video Technology*, 12(12), December 2002;
  - for audio, see **Performance Evaluation of Real-time Speech Through a Packet Network: a Random Neural Network Based Approach**, S. Mohamed, G. Rubino, M. Varela, *Performance Evaluation*, 57(2):141–162, 2003.
- For performance evaluation aspects (coupling PSQA with traditional modeling techniques), **A new approach for the prediction of end-to-end performance of multimedia streams**, G. Rubino and M. Varela, in the proceedings of the IEEE Int. Conf. *QEST'2004*, Twente, September 2004.

Some papers (cont'd)

- For a global view of PSQA and some new material about the learning tool used, see **Quantifying the Quality of Audio and Video Transmissions over the Internet: the PSQA Approach**, G. Rubino, in *Design and Operations of Communication Networks: A Review of Wired and Wireless Modelling and Management Challenges*, Imperial College Press, Ed.: J. Barria, 2005.

Some papers (cont'd)

- For applications to wireless connections in a home network environment, see **Controlling Multimedia QoS in the Future Home Network Using the PSQA Metric**, The Computer Journal 2006 49: 137-155, March 2006.
- For other details including a comparison between PSQA and other learning tools, see **Evaluating users' satisfaction in packet networks using Random Neural Networks**, G. Rubino, P. Tirilly and M. Varela, invited in *ICANN 2006: International Conference on Artificial Neural Networks*, Athens, Greece, September 2006.

Current work

- As already said, we are currently working on an application of PSQA to network design. Target: P2P networks (with central control) for transmission of live video.
  Recently, experimental work started on PlanetLab.
- We are also working in different projects with important participation of industrial partners, where the PSQA technology is being used:
  - a large French project about P2P design:
    - PSQA used to evaluate different algorithmic proposals;
  - a large French project about video transmission networks with mobile terminals:
    - the main goal is the design of a system for network diagnostic,
    - PSQA modules will be embedded in mobile terminals;
  - an European project on the impact of SVC (scalable coding) on QoE:
    - PSQA will be used to analyze the impact of different factors associated with SVC codecs on the QoE.
- In all three projects the design of control systems taken advantage of the capacity of PSQA to work in real time will be explored.

Papers on the P2P project

- P. Rodríguez-Bocca, H. Cancela, and G. Rubino. **Perceptual quality in P2P video streaming policies**, GLOBECOM'07.
- P. Rodríguez-Bocca, H. Cancela, and G. Rubino. **Video Quality Guarantees in Multi-Source Streaming Techniques**, LANC'07. Best paper award by ACM-SIGCOMM.
- H. Cancela, F. Robledo Amoza, P. Rodríguez-Bocca, G. Rubino, and A. Sabiguero. **A robust P2P streaming architecture and its application to a high quality live-video service**, Electronic Notes in Discrete Mathematics (and LAGOS'07).
- P. Rodríguez-Bocca, G. Rubino, and L. Stábile. **Multi-Source Video Streaming Suite**, IPOM'07.
- A.P. Couto da Silva, P. Rodríguez-Bocca, and G. Rubino. **Optimal quality?of?experience design for a p2p multi-source video streaming**, ICC'08.

Mid-term perspectives

For the close future:

- We are developing a new generation of PSQA tools, with the following properties:
  - more accurate, more robust and faster:
    - key idea: including cumulated knowledge into the learning tool
    - second idea: combining different techniques explored in previous works
    - also using algorithmic ideas coming from other fields increasing the learning capabilities of the tools and the convergence speed
  - capability of solving the inverse problem: find the region of the parameters such that quality is at least Q_0;
- First steps towards industrialization are being explored with some of the present industrial partners.
- More on a long-term basis: we are studying the coupling of PSQA with predicting tools: the idea is to anticipate the quality level in the next Δ units of time, for controlling applications.

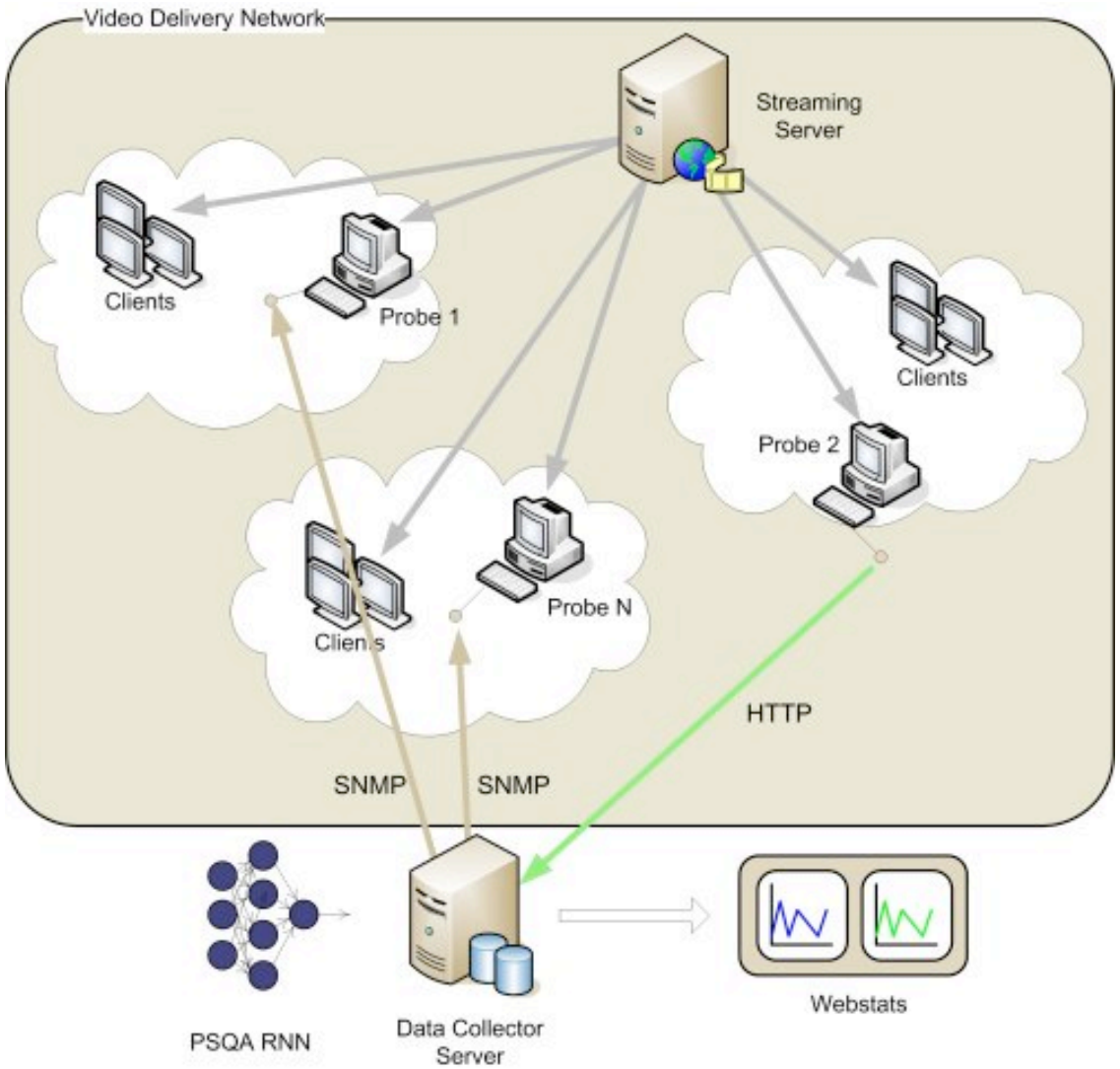
Network audit

We are also working on the prototype of a network monitoring system:

- some experiments run on PlanetLab
- a telco is starting to use it
- a first illustrative prototype will be demonstrated next week in Sigmetrics'2008.

- Build a sample of the population of terminals of some specific network (or family of networks) sending data through the Internet.
- Put a PSQA module at each of these points, capable of sending reports to some central processing center.
- On the data sent by the different measuring points, build an estimator of
  - the instantaneous average perceived quality of the network (that is, a sort of $\mathrm{E}(Q(t))$)
  - the proportion of users receiving a quality level higher than some minimum acceptable value
  - ...

The architecture of the monitoring system



A "graphical" overview