Joint End-to-End Loss–Delay Hidden Markov Model for Periodic UDP Traffic Over the Internet

Pierluigi Salvo Rossi, Gianmarco Romano, Francesco Palmieri, and Giulio Iannello

***Abstract*—Performance of real-time applications on network communication channels is strongly related to losses and temporal delays. Several studies showed that these network features may be correlated and exhibit a certain degree of memory such as bursty losses and delays. The memory and the statistical dependence between losses and temporal delays suggest that the channel may be well modeled by a hidden Markov model (HMM) with appropriate hidden variables that capture the current state of the network. In this paper, an HMM is proposed that shows excellent performance in modeling typical channel behaviors in a set of real packet links. The system is trained with a modified version of the Expectation-Maximization (EM) algorithm. Hidden-state analysis shows how the state variables characterize channel dynamics. State-sequence estimation is obtained by the use of Viterbi algorithm. Real-time modeling of the channel is the first step to implement adaptive communication strategies.**

***Index Terms*—Hidden Markov models (HMM), loss-and-delay analysis, network models.**

I. Introduction

COMMUNICATING over the global network will continue to characterize the coming years with growth in connectivity and in types of data links. Wireless networks, satellites, cable modems, routers and all kinds of communication devices will constitute the very heterogeneous makeup of most connections. Moreover, communication often happens with very little control over congestion and information accuracy.

In real-time communications, several strategies have been proposed in the literature to improve the performance of transport protocols like Transmission Control Protocol (TCP) and User Datagram Protocol (UDP), e.g., forward-error correction (FEC) strategies based on the use of robust transforms [14] or redundant codes. These strategies are more or less tuned to the source, but performance assessment is difficult unless a reasonable channel model is available. Usually a very simple lossy channel is assumed without any specific reference to time dependence or time variations.

Manuscript received September 15, 2003; revised March 6, 2005. This work was supported in part by the Ministero dell'Istruzione, dell'Universitáe della Ricerca (MIUR) in the framework of the FIRB Project "Middleware for advanced services over large-scale, wired-wireless distributed systems (WEB-MINDS)." The associate editor coordinating the review of this paper and approving it for publication is Dr. Xiaodong Wang.

P. Salvo Rossi is with the Dipartimento di Informatica e Sistemistica, Università di Napoli "Federico II," 80125 Napoli, Italy (e-mail: salvoros@unina.it).

G. Romano and F. Palmieri are with the Dipartimento di Ingegneria dell'Informazione, Seconda Università di Napoli, 81031 Aversa (CE), Italy (e-mail: gianmarco.romano@unina2.it; francesco.palmieri@unina2.it).

G. Iannello is with the Universit̀ Campus Bio-Medico di Roma, 00155 Roma, Italy (e-mail: g.iannello@unicampus.it).

Digital Object Identifier 10.1109/TSP.2005.861066

Our work attempts to build a model framework for a generic end-to-end connection. Too many details would be necessary to keep into account congestion, protocols, lossy links, noisy channels, etcetera. However, a global equivalent end-to-end model that could be easily estimated via low-speed two-way connections seems to be viable. Since most network connections from the MAC-layer up on the OSI stack are based on packets, such a mode will be our reference framework: packets are loaded at the transmitter and may arrive at the receiver after a delay, or be totally lost. In real-time communication packets that arrive after a maximum delay could also be considered lost. In addition, packets that contain errors are usually either corrected or disregarded and are therefore not considered in our modeling.

We want to model the system in a way that carries information on current congestion that may determine variable loss rates and average delays. Loss phenomena often show bursty behavior, i.e., they cannot be thought of as a memoryless stochastic process. Several works [7], [11], [12] showed that temporal delays seem to be not well modeled as independent identically distributed (i.i.d.) random variables, but present a certain degree of memory. In real communication networks, losses and delays are strongly correlated: It has been observed [12] that in proximity of a loss, larger delays tend to occur.

The works of Gilbert and Elliott [1], [2] on modeling burst-error channels for bit transmission showed how a simple two-state hidden Markov model (HMM) was effective in characterizing some real communication channels. As in the case of bit-transmission channels, end-to-end packet channels show burst–loss behavior. Jiang and Schulzrinne [12] investigated lossy behavior of packet channels, finding that a Markov model is not able to appropriately describe channel interloss behavior. They also found that delays manifest temporal dependency, i.e., they should not be assumed to be a memoryless phenomenon. Salamatian and Vaton [13] found that an HMM trained with experimental data seems to capture channel loss behavior and found that an HMM with two to four hidden states fits well with experimental data. Liu, Matta, and Crovella [15] used an HMM-based loss-delay modeling of TCP traffic in order to infer losses nature in hybrid wired/wireless environments. They found that such a kind of modeling can be used to control TCP congestion avoidance mechanism. Similar works have been done by Turin, Sondhi, Zorzi *et al.*, and van Nobelen [6], [8], [10] on error sources for digital channels and wireless fading links.

These works suggest that a Bayesian state-conditioned model may be effective in capturing the dynamic behavior of losses and delays on end-to-end packet channels [18], [19], [22]. The definition of such a model is highly desirable for designing and

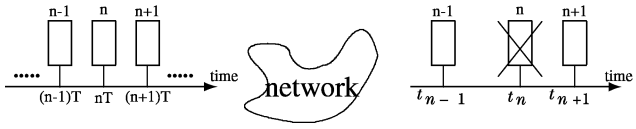

Fig. 1. End-to-end packet channel.

evaluating coding strategies. Furthermore, the possibility of learning online the parameters of the model opens the way to design efficient content-adaptive services. Hidden states of the model represent different working conditions of the channel. Current state knowledge and prediction of state transitions may enable a powerful characterization of future channel behavior, which could be used to implement content-adaptive strategies for coding (e.g., multiple description coding) and scheduling (e.g., traffic shaping).

In this paper, we propose a comprehensive model that jointly describes losses and delays. The model is an HMM trained with a version of the Expectation–Maximization (EM) algorithm to capture channel dynamics. A set of simulations based on measurements obtained on real packet links confirms how effective the model is and how hidden states (automatically found) can be significant.

The rest of the paper is organized as follows. In Section II, we describe the proposed model; the algorithm for finding the model parameters is presented in Section III. In Section IV, we present how the model works on real communication channels, specifically in Sections IV-A-1) and IV-A-2) we describe, respectively, the training step and the capability of generalization of the model. Some conclusive remarks are given in Section V.

II. The Model

Our reference model is shown in Fig. 1: a periodic source traffic with interdeparture period T and fixed packet size of N_b bits. The system data rate is $R = N_b/T$ b/s. The network randomly cancels and delays packets according to current congestion. Transmitted packets are numbered $n = 1, 2, \ldots$; t_n and τ_n are the arrival time and the accumulated delay of the nth packet, respectively, i.e., $\tau_n = t_n - nT$.

Memory and correlation presence in loss-and-delay dynamics of the communication channels suggest the introduction of a hidden-state variable carrying information about link congestion. The state variable stochastically influences losses and delays. It is hidden because our knowledge about it can only be inferred from observation of losses and delays, and there is no way to access it directly. We will see later how the learning algorithm automatically "discovers" the hidden states and how they can be associated to various levels of congestion. Since we assume the packet rate to be constant, our model is synchronous to departure time.

Let us denote x_n the state of the link at time step n, with $x_n \in \{s_1, s_2, \ldots, s_N\}$, where s_i is the ith state, among N possible ones. τ_n is the delay for the nth packet, and $l_n \in \{v_1, v_2\}$ is a binary variable where v_1 and v_2 correspond, respectively, to absence or presence of a loss. It should be noted that in the presence of a loss, the delay has no real value. Such an event could be associated to infinite delay, but for better modeling we distinguish the situation with $\tau_n = \infty$ from the lost packet, and we attribute to lost packets the fictitious negative value $\tau_n = -1$, i.e., $\begin{cases} \tau_n | \{l_n = v_1\} \in [0, +\infty) \\ \tau_n | \{l_n = v_2\} = -1 \end{cases}$.

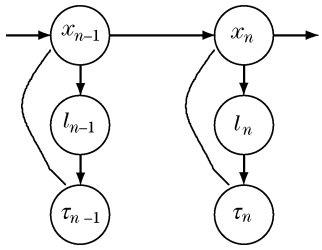

Fig. 2. Bayesian model for packet channel.

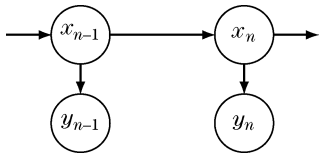

Fig. 3. Hidden Markov model.

Our reference Bayesian model is shown in Fig. 2, where the arrows represent statistical dependence among variables. More specifically, the set of parameters characterizing the model is $\Lambda = \{\mathbf{A}, \mathbf{p}, f_1(\tau), f_2(\tau), \ldots, f_N(\tau)\}$, defined as follows:

- $\mathbf{A}$ is the state transition matrix, i.e.,

$$\mathbf{A} = [a_{ij}] = [\Pr(x_{n+1} = s_j | x_n = s_i)]$$

- $\mathbf{p} = (p_1, p_2, \ldots, p_N)$ is the state-conditioned loss probability vector, i.e.,

$$\begin{cases} p_i = \Pr(l_n = v_1 | x_n = s_i) \\ 1 - p_i = \Pr(l_n = v_2 | x_n = s_i) \end{cases}$$

- $f_i(\tau)$ is the state-conditioned delay probability density function (pdf), i.e.,

$$\Pr(\tau_n > t | x_n = s_i, l_n = v_1) = \int_t^{+\infty} f_i(\tau) d\tau.$$

The model can be reduced to an HMM, as seen in Fig. 3, with a hidden variable x_n and an observable variable y_n that represents jointly losses and delays as $y_n = \begin{cases} \tau_n \; if \; l_n = v_1 \\ -1 \; if \; l_n = v_2 \end{cases}$, summarized as follows:

- x_n is a discrete random variable whose dynamic behavior is governed by the transition matrix $\mathbf{A}$;
- y_n is a hybrid random variable that, given $\{x_n = s_i\}$, is characterized by the mixed pdf

$$b_i(t) = p_i f_i(t) + (1 - p_i)\delta(t + 1). \tag{1}$$

Shown in Fig. 4, y_n is a hybrid variable obtained as a mixture of two components (one continuous, the other discrete), where there is a probability mass concentrated in -1 to model losses.

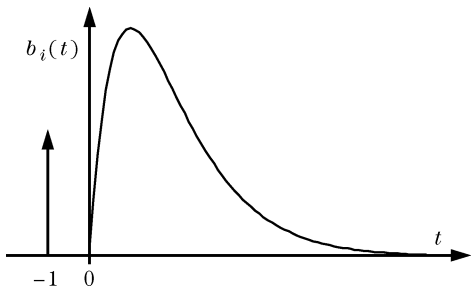

Fig. 4. State-conditioned pdf for the hybrid variable.

This allows a compact representation with nonoverlapping distributions. The continuous component describes network delay behavior in the absence of losses, whereas the discrete component describes losses.

If $\pi = (\pi_1, \pi_2, \ldots, \pi_N)$ is the steady-state probability distribution, i.e., $\pi_i = \lim_{n\to\infty}\{\Pr(x_n = s_i)\}$, the average loss probability and the average delay of the model are

$$P_{\text{loss}} = \sum_{i=1}^{N} \pi_i (1 - p_i), \quad D_{\text{mean}} = \sum_{i=1}^{N} \pi_i d_i \tag{2}$$

where $d_i = \int_0^{+\infty} t f_i(t) dt$ is the state-conditioned mean delay.

III. Learning the Model Parameters

The model, proposed to account for the time-varying nature of the channel behavior, would be just speculative if we did not have a technique for learning the model parameters from online measurements. Furthermore, such a model would be useful only if the channel holds on the same stochastic behavior observed during the training step, i.e., the channel alternates different short-term behavior showing a long-term stationarity. In this case, the model has generalization capability and is useful for data transmission characterization. Although in a real scenario the channel stationarity cannot be assumed *a priori* with certainty, we assume that stationarity holds for a time interval much longer than the time needed to perform the model training. The experiments we performed, described in Section IV, seem to confirm that this assumption is realistic in practice, showing that a valid data window for the training procedure is less than 10% of the stationarity window (more precisely, in the examples in Sections IV-A and IV-B, it is 3% and 8%, respectively).

The EM algorithm [9] is an optimization procedure that allows learning of a new set of parameters for a stochastic model according to improvements of the likelihood of a given sequence of observable variables. For structures like the HMM of Fig. 3 this optimization technique reduces to the Forward–Backward algorithm [3]–[5] studied for discrete and continuous observable variables with a broad class of allowed state-conditioned pdf's.

More specifically, given a sequence of observable variables $\mathbf{y} = (y_1, y_2, \ldots, y_K)$, referred to as the *training sequence*, we want to find the set of parameters Λ_0 such that the likelihood of the training sequence is maximum, i.e.,

$$\begin{cases} L(\mathbf{y};\Lambda) = \Pr(\mathbf{y}|\Lambda) \\ \Lambda_0 = \arg\max_{\Lambda}\{L(\mathbf{y};\Lambda)\} \end{cases}.$$

The Forward–Backward algorithm is an iterative procedure looking for a local maximum of the likelihood function, which typically depends on the starting point Λ. When necessary, repeated starts with different initial conditions provide the global solution. The algorithm works applying iteratively a transformation $\hat{\Lambda} = T(\mathbf{y};\Lambda)$ such that $L(\mathbf{y};\hat{\Lambda}) \geq L(\mathbf{y};\Lambda)$.

Several works [7], [11], [16] showed that delay statistics on real Internet paths fit well a shifted Gamma shape. Our choice of state-conditioned pdf's for modeling delays is a Gamma distribution with only two parameters, in order to have a good "fitting freedom" and keep the number of parameters to be estimated low, as follows:

$$f_i(t) = \frac{\left(\frac{t}{\vartheta_i}\right)^{\gamma_i - 1} e^{-\left(\frac{t}{\vartheta_i}\right)}}{\vartheta_i \Gamma(\gamma_i)} t > 0.$$

Therefore, $\Lambda = \{\mathbf{A}, \mathbf{p}, \gamma, \vartheta\}$, where $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_N)$ and $\vartheta = (\vartheta_1, \vartheta_2, \ldots, \vartheta_N)$ or equivalently $\Lambda = \{\mathbf{A}, \mathbf{p}, \mathbf{d}, \sigma\}$, where $\mathbf{d} = (d_1, d_2, \ldots, d_N)$ and $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_N)$ denote state-conditioned delay means and standard deviations, respectively, i.e., from (3)

$$d_i = E[\tau_n | \{x_n = s_i\}] = \gamma_i \vartheta_i,$$
$$\sigma_i^2 = Var[\tau_n | \{x_n = s_i\}] = \int (t - d_i)^2 f_i(t) dt = \gamma_i \vartheta_i^2.$$

The algorithm works as follows.

1) `Set input parameters`: This includes the training sequence $\mathbf{y} = (y_1, y_2, \ldots, y_K)$, the number of states N, and the stopping parameter ε.
2) `Initialization step`: The algorithm is initialized by a symmetric model, which we will refer to as the *starting model*, i.e.,

$$\begin{cases} a_{ij} = \frac{1}{N} \\ p_i = 0.5 \\ \gamma_i = \frac{i\tau_{\max}}{(N+1)} \\ \vartheta_i = 1 \end{cases}$$

where $\tau_{\max} = \max\{y_n\}$ is the maximum measured delay. Other initializations are possible. We have adopted this one for simplicity and to provide a starting point that is not too far from the solution.
3) `Iterative step`: The transformation $\hat{\Lambda} = T(\mathbf{y};\Lambda)$ is iteratively applied until the stopping condition $L(\mathbf{y};\hat{\Lambda}) - L(\mathbf{y};\Lambda) \leq \varepsilon$ is verified.

The transformation $T(.;.)$ is based on computation of the forward and backward partial likelihoods, respectively

$$\alpha_k(j) = \sum_{i=1}^{N} \alpha_{k-1}(i) a_{ij} b_j(y_k),$$
$$\beta_k(i) = \sum_{j=1}^{N} a_{ij} b_j(y_{k+1}) \beta_{k+1}(j)$$

where $\alpha_1(j) = \delta_{1,j}$ and $\beta_K(i) = 1$. More specifically, denote

$$\rho_k(j) = \sum_{i=1}^{N} \alpha_{k-1}(i) a_{ij} p_j \left.\frac{\partial b_j(t)}{\partial p_j}\right|_{t=y_k}$$

which in our case becomes

$$\rho_k(j) = \sum_{i=1}^{N} \alpha_{k-1}(i) a_{ij} p_j \left[f_j(y_k) - \delta(y_k + 1) \right]. \quad (3)$$

The update $\hat{\Lambda} = \{\hat{\mathbf{A}}, \hat{\mathbf{p}}, \hat{\mathbf{d}}, \hat{\sigma}\}$ is based on the following equations:

$$\hat{a}_{ij} = \frac{\sum_{k=1}^{K-1} \alpha_k(i) a_{ij} b_j(y_{k+1}) \beta_{k+1}(j)}{\sum_{k=1}^{K-1} \alpha_k(i) \beta_k(i)}$$

$$\hat{p}_i = \frac{\sum_{k=1}^{K} \rho_k(i) \beta_k(i)}{\sum_{k=1}^{K} \alpha_k(i) \beta_k(i)}$$

$$\hat{d}_i = \frac{\sum_{k=1}^{K} \rho_k(i) \beta_k(i) y_k}{\sum_{k=1}^{K} \rho_k(i) \beta_k(i)}$$

$$\hat{\sigma}_i^2 = \frac{\sum_{k=1}^{K} \rho_k(i) \beta_k(i) (y_k - d_i)^2}{\sum_{k=1}^{K} \rho_k(i) \beta_k(i)}.$$

The problem of the Dirac-impulse in the state-conditioned pdf ((1) and (3)) is avoided by replacing the hybrid pdf $b_i(t)$ with the following modified function:

$$\tilde{b}_i(t) = p_i f_i(t) + (1 - p_i) g(t)$$

where $g(t)$ is any pdf such that $g(t) = 0, \forall t \geq 0$ to avoid overlapping supports between $f_i(t)$ and $g(t)$. Obviously, while the set $\{f_i(t)\}_{i=1}^{N}$ will be adjusted by the iterative procedure, $g(t)$ will remain unchanged, as only its area is relevant. This means that losses in the algorithm can be randomized according to $g(t)$. In our algorithm, losses are simply randomized according to a narrow uniform distribution around -1, i.e.,

$$g(t) = \begin{cases} \frac{1}{2\Delta}, & t \in [-1 - \Delta, -1 + \Delta] \\ 0, & \text{otherwise} \end{cases}$$

where Δ is an arbitrary small number. The likelihood is evaluated by $L(\mathbf{y}; \Lambda) = \sum_{i=1}^{N} \alpha_K(i)$.

It must be remembered that the implementation of the algorithm requires proper scaling to avoid underflow [5].

IV. Experimental Results

Measures of losses and delays have been performed on real Internet channels (among Dipartimento di Informatica e Sistemistica, Università di Napoli "Federico II"; Dipartimento di Ingegneria dell'Informazione, Seconda Università di Napoli; Università Campus Bio-Medico di Roma), using the software Distributed-Internet Traffic Generator (D-ITG) [20].

D-ITG (a new version of Mtools [17]) can generate both traffic at transport layer and "layer 4–7." It implements both TCP and UDP traffic generation according to several statistical distributions both for interdeparture times and packet sizes. D-ITG allows measurement collection from complex traffic scenarios furnishing information about transmitted and received packets. D-ITG was used to obtain loss–delay sequences of synthetic periodic UDP traffic sent on real Internet channels with background load present. We do not address the problem of packet-delay estimation assuming that the accuracy of measurements is acceptable. A little portion of the sequences was used as the training sequence to learn model parameters. Performances of trained model are tested on the remaining portions of sequences. A portion of a loss–delay sequence is shown in Fig. 5. Losses are indicated with negative values of y_n.

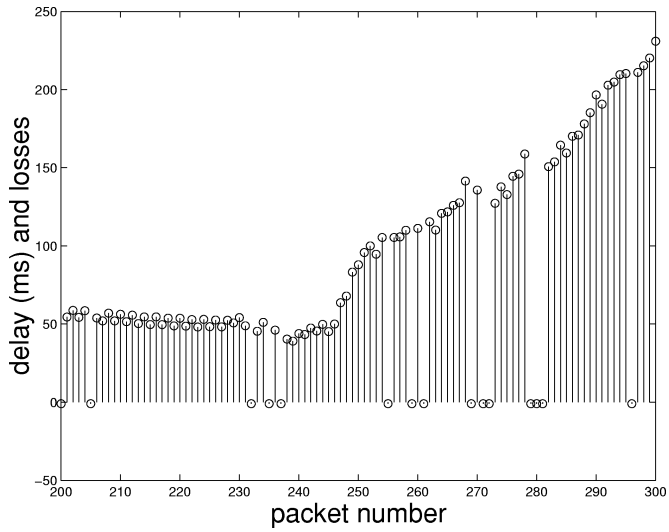

Fig. 5. Portion of a measured trace on a real network.

In the following, two examples are described. The experiments reported here are not to be considered exhaustive, but they suggest a very typical scenario in which losses and delays show the usual "burstiness."

A. Example I

The following example was performed in April 2003 on the path between Dipartimento di Informatica e Sistemistica, Università di Napoli "Federico II," and Dipartimento di Ingegneria dell'Informazione, Seconda Università di Napoli, in which the interdeparture period is $T = 5$ ms, and the packet size is $N_b = 8000$ b, $(R = 1.6$ Mb/s$)$. The Internet path was composed of six wired links (obtained by the use of `traceroute`). The path has a bottleneck link of 2 Mb/s (nominal bandwidth). The training step and the test step are described, respectively, in Sections IV-A-1) and IV-A-2), while Sections IV-A-3) shows the performance of the trained models when considered as channel simulators.

1) Training: Fig. 6 shows a typical trend of log-likelihood evolution during our EM learning procedure. We have used a training sequence of $K = 1000$ samples, which was used to train models with two, three, and four states. The algorithm convergence is reached after a few iterations, although very small values for the likelihood are obtained.[1] In our experiments, 10 to 20 iterations are enough to reach convergence such that the trained model guarantees sufficient generalization capability. The complexity of the algorithm is linear with the length of the training sequence [5]. Simulations with a Matlab implementation of the algorithm run on a 600-MHz Athlon 4 processor took approximately 15 s.

[1]The likelihood can be thought as the probability to generate with a Monte Carlo simulation the training sequence (1000 samples length). The scaling procedure, mentioned in Section III, has to be implemented in order to manage such very small numbers whose computation exceeds the precision range of any machine [5].

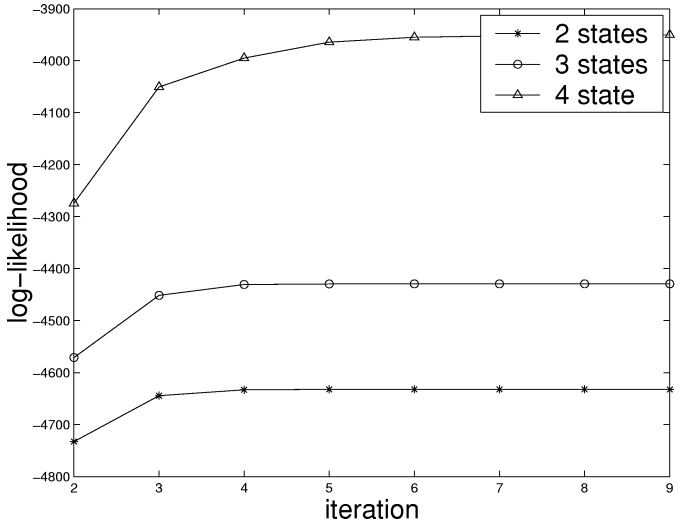

Fig. 6. Log-likelihood trend in the learning procedure (Example I).

TABLE I
AVERAGE AND TRAINING STATISTICS COMPARISON (EXAMPLE I)

	training	starting	2-state	3-state	4-state
P_{loss}	0.117	0.500	0.132	0.132	0.133
D_{mean} (ms)	206.10	286.90	246.95	253.64	272.42

It is worth noting that the length of the training sequence individuates a tradeoff for the computational complexity. In Section III, we said that the model parameters learned during the training step are useful within the stationarity horizon of the channel, and this will be clarified in Section IV-A-2). However, in a real scenario, where an adaptive mechanism has to be considered, the model parameters have to be relearned periodically as the network environment can change dramatically. It is evident that reducing the length of the training sequence reduces the complexity of the training algorithm, but it also results in a stationarity horizon that is too short (e.g., only one a few modes can be observed), meaning that the training step has to be run more frequently. In this paper, we do not address the problem of the optimum choice of the training sequence.

Table I summarizes the results of the learning procedure in terms of the average statistics (see (2)), showing how the trained models capture average statistics. It is worth noting that the average statistics do not necessarily match the ones of the training sequence with very high accuracy. The trained models, even in the case when high accuracy is not reached, match appropriately the histogram of the training sequence. Furthermore, the trained models mainly try to capture the statistical dependence existing between loss and delays such as their memory and correlation, as will be shown in the following.

Tables II–IV show the steady-state probability (π_i), the loss probability $(1-p_i)$, and the average delay (d_i) for the ith state. They confirm an intuitive interpretation of the state that will be more evident later, when the evolution of the state variable is examined.

Fig. 7 shows the delay pdf's before and after learning with two-, three-, and four-state models in comparison to a delay histogram. The histogram clearly shows a multimodal behavior. It is encouraging to see how well the model captures at increasing resolution the measured delay statistics as the number of hidden states is increased.

TABLE II
STATE-CONDITIONED STATISTICS FOR A TWO-STATE MODEL (EXAMPLE I)

	π_1 $1-p_1$ d_1 (ms)	π_2 $1-p_2$ d_2 (ms)
starting	0.500 0.500 201.14	0.500 0.500 372.65
trained	0.450 0.085 131.14	0.550 0.170 341.69

TABLE III
STATE-CONDITIONED STATISTICS FOR A THREE-STATE MODEL (EXAMPLE I)

	π_1 $1-p_1$ d_1 (ms)	π_2 $1-p_2$ d_2 (ms)	π_3 $1-p_3$ d_3 (ms)
starting	0.333 0.500 158.27	0.333 0.500 286.90	0.333 0.500 415.53
trained	0.441 0.085 130.72	0.383 0.111 290.62	0.176 0.298 481.13

TABLE IV
STATE-CONDITIONED STATISTICS FOR A FOUR-STATE MODEL (EXAMPLE I)

	π_1 $1-p_1$ d_1 (ms)	π_2 $1-p_2$ d_2 (ms)	π_3 $1-p_3$ d_3 (ms)	π_4 $1-p_4$ d_4 (ms)
starting	0.250 0.500 132.54	0.250 0.500 235.45	0.250 0.500 338.35	0.250 0.500 441.26
trained	0.128 0.139 77.98	0.339 0.045 181.98	0.330 0.120 312.53	0.203 0.297 480.87

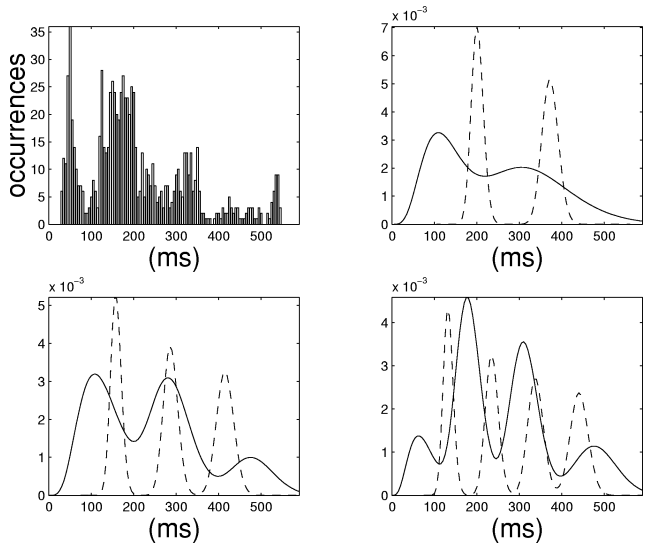

Fig. 7. Learning delays statistics (Example I): histogram of measured delays (the bin size is 5 ms) and continues term of pdf of two-, three-, and four-state starting (dashed line) and trained (solid line) models, respectively.

Hence, to verify how the hidden-state variable x_n captures the current channel congestion, a Viterbi algorithm [5] has been applied to the training sequence. We note again that the Viterbi algorithm furnishes the most likely state sequence

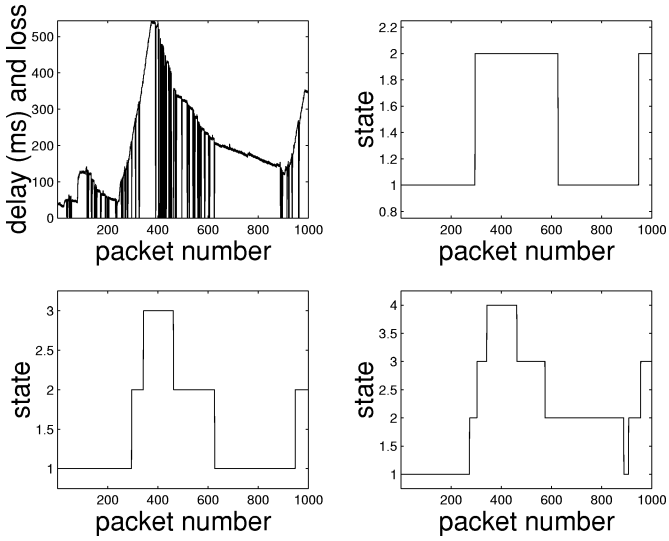

Fig. 8. State-sequence estimation (Example I): training sequence and two-, three-, and four-state trained models, respectively.

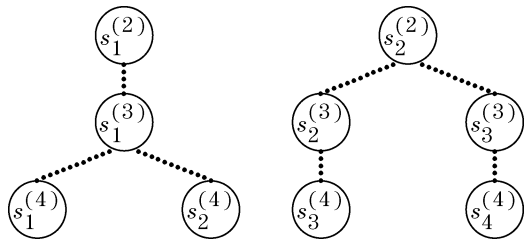

Fig. 9. Correspondences among states of the trained models (Example I).

$\mathbf{x} = (x_1, x_2, \ldots, x_K)$, i.e., the state sequence such that the *a posteriori* probability $\Pr(\mathbf{x}/\mathbf{y}, \Lambda)$ is maximum.

Fig. 8 shows the temporal evolution of the training sequence $\mathbf{y} = (y_1, y_2, \ldots, y_K)$ and the state sequences obtained by use of the Viterbi algorithm on the previous two-, three-, and four-state trained models. The trained models give state sequences that follow loss–delay network behavior quite well.

With reference to Fig. 8, we would now like to furnish a qualitative interpretation of states automatically found. If $s_i^{(k)}$ is the ith state of the k-state model, then the following occurs:

- the two-state model emerges to distinguish two situations: $s_1^{(2)}$ for lower delays and fewer losses and $s_2^{(2)}$ for large delays and many losses;
- the three-state model seems to use its states to distinguish the same two situations as the previous model with $s_1^{(3)}$ resembling $s_1^{(2)}$, while $s_2^{(2)}$ is now split in two states: $s_2^{(3)}$ for many losses and $s_3^{(3)}$ describing very high-delays situation;
- the four-state model distinguishes the same two situations as the two-state model, but now each one of them is described with two states: $s_3^{(4)}$ and $s_4^{(4)}$, respectively, corresponding to $s_2^{(3)}$ and $s_3^{(3)}$ for the high-delays/many-losses situation, while for the low-delays situation $s_1^{(2)}$ or $s_1^{(3)}$ too is split: $s_1^{(4)}$ describes low-delays and losses and $s_2^{(4)}$ describes low delays and very few losses.

Fig. 9 synthesizes the correspondences we noted about states of the trained models, i.e., the relations among the states of different trained models related to the considered example (the dotted lines highlight the corresponding states of different models). Let us denote $\pi_i^{(k)}$, $1 - p_i^{(k)}$, and $d_i^{(k)}$ as the steady-state probability, the loss probability, and the average delay of the state $s_i^{(k)}$, respectively. Let $P_{\text{loss,a}}^{(k)}$, $D_{\text{mean,a}}^{(k)}$ and $P_{\text{loss,b}}^{(k)}$, $D_{\text{mean,b}}^{(k)}$ be the loss probabilities and the average delays in the two situations previously inferred (low delays or few losses, and larger delays or many losses) for the k-state model; $\pi_{\text{a}}^{(k)}$ and $\pi_{\text{b}}^{(k)}$ are the corresponding steady probabilities.

From Tables II–IV, the following equalities strengthen the effectiveness of the various models, confirming the correspondence, previously described, among hidden states as well as the significance of the state variable x_n.

$$\begin{cases} \pi_{\text{a}}^{(2)} = \pi_1^{(2)} = 0.450 \\ \pi_{\text{a}}^{(3)} = \pi_1^{(3)} = 0.441 \\ \pi_{\text{a}}^{(4)} = \pi_1^{(4)} + \pi_2^{(4)} = 0.467 \end{cases}$$

$$\begin{cases} \pi_{\text{b}}^{(2)} = \pi_2^{(2)} = 0.550 \\ \pi_{\text{b}}^{(3)} = \pi_2^{(3)} + \pi_3^{(3)} = 0.559 \\ \pi_{\text{b}}^{(4)} = \pi_3^{(4)} + \pi_4^{(4)} = 0.533 \end{cases}$$

$$\begin{cases} P_{\text{loss,a}}^{(2)} = \pi_1^{(2)}(1 - p_1^{(2)}) = 0.038 \\ P_{\text{loss,a}}^{(3)} = \pi_1^{(3)}(1 - p_1^{(3)}) = 0.037 \\ P_{\text{loss,a}}^{(4)} = \pi_1^{(4)}(1 - p_1^{(4)}) + \pi_2^{(4)}(1 - p_2^{(4)}) = 0.033 \end{cases}$$

$$\begin{cases} P_{\text{loss,b}}^{(2)} = \pi_2^{(2)}(1 - p_2^{(2)}) = 0.094 \\ P_{\text{loss,b}}^{(3)} = \pi_2^{(3)}(1 - p_2^{(3)}) + \pi_3^{(3)}(1 - p_3^{(3)}) = 0.095 \\ P_{\text{loss,b}}^{(4)} = \pi_3^{(4)}(1 - p_3^{(4)}) + \pi_4^{(4)}(1 - p_4^{(4)}) = 0.100 \end{cases}$$

$$\begin{cases} D_{\text{mean,a}}^{(2)} = d_1^{(2)} = 131.14 \text{ ms} \\ D_{\text{mean,a}}^{(3)} = d_1^{(3)} = 130.72 \text{ ms} \\ D_{\text{mean,a}}^{(4)} = \frac{\pi_1^{(4)}}{\pi_1^{(4)} + \pi_2^{(4)}} d_1^{(4)} + \frac{\pi_2^{(4)}}{\pi_1^{(4)} + \pi_2^{(4)}} d_2^{(4)} = 153.52 \text{ ms} \end{cases}$$

$$\begin{cases} D_{\text{mean,b}}^{(2)} = d_2^{(2)} = 341.69 \text{ ms} \\ D_{\text{mean,b}}^{(3)} = \frac{\pi_2^{(3)}}{\pi_2^{(3)} + \pi_3^{(3)}} d_2^{(3)} + \frac{\pi_3^{(3)}}{\pi_2^{(3)} + \pi_3^{(3)}} d_3^{(3)} = 350.65 \text{ ms} \\ D_{\text{mean,b}}^{(4)} = \frac{\pi_3^{(4)}}{\pi_3^{(4)} + \pi_4^{(4)}} d_3^{(4)} + \frac{\pi_4^{(4)}}{\pi_3^{(4)} + \pi_4^{(4)}} d_4^{(4)} = 376.55 \text{ ms} \end{cases}$$

where $D_{\text{mean},\xi}^{(k)} = \sum_{i \in \xi} \Pr(s_i^{(k)}|\xi) d_i^{(k)} = \sum_{i \in \xi} (\pi_i^{(k)}/\pi_\xi^{(k)}) d_i^{(k)}$ and where $\xi = \{\text{a}, \text{b}\}$ and $k = \{2, 3, 4\}$.

It is worth noting that the different modes found by the models (especially in the four-state model) denotes different local behavior in terms of loss and delays. This means that in a TCP scenario, the model could be used to distinguish between congestion losses and error losses. Modification of the model in order to work in a TCP scenario is an aspect that interests us.

2) Model Generalization: A trained model, to be useful, has to be tested on data that was not seen during training. Such generalization property has been verified for our model as it matches also the future behavior of the channel.

Fig. 10 shows the log-likelihood of the previous two-, three-, and four-state trained models. D-ITG was used to obtain a loss–delay sequence of length 31 000, where the first $K = 1000$ samples constitute the training sequence. The remaining 30 000 samples were grouped in blocks of $K = 1000$ consecutive samples. Each block represents an element for the test set, i.e., a *test sequence*.

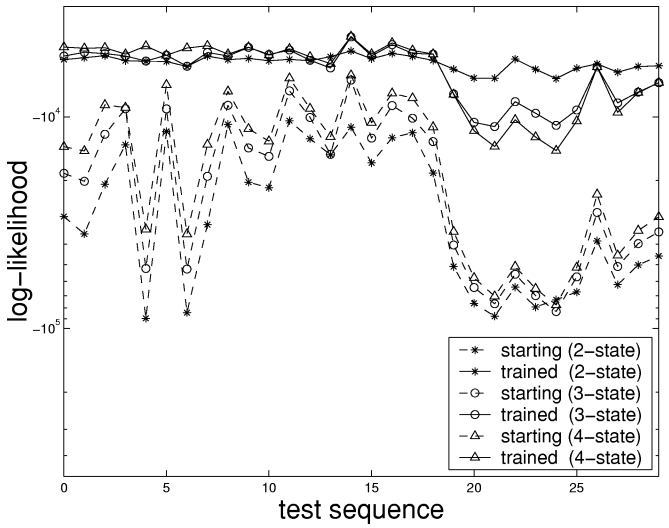

Fig. 10. Capacity of generalization of the model by means of the log-likelihood (Example I).

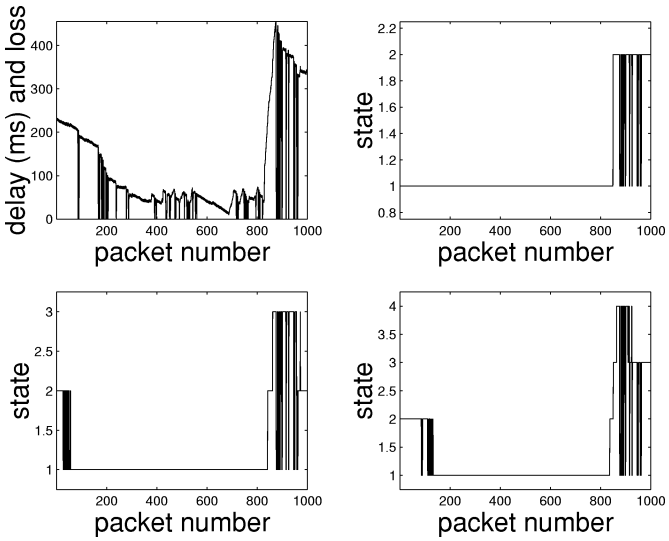

Fig. 11. State-sequence estimation before learning (sequence $n.\ 6$, Example I): test sequence and two-, three-, and four-state starting models, respectively.

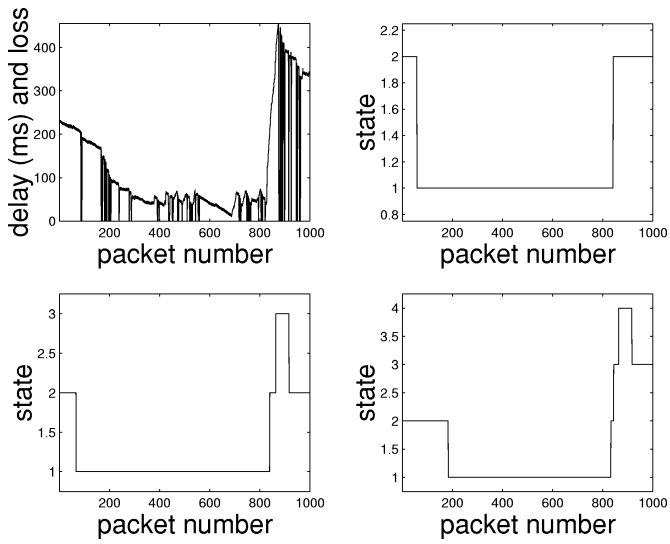

Fig. 12. State-sequence estimation after learning (sequence $n.\ 6$, Example I): test sequence and two-, three-, and four-state trained models, respectively.

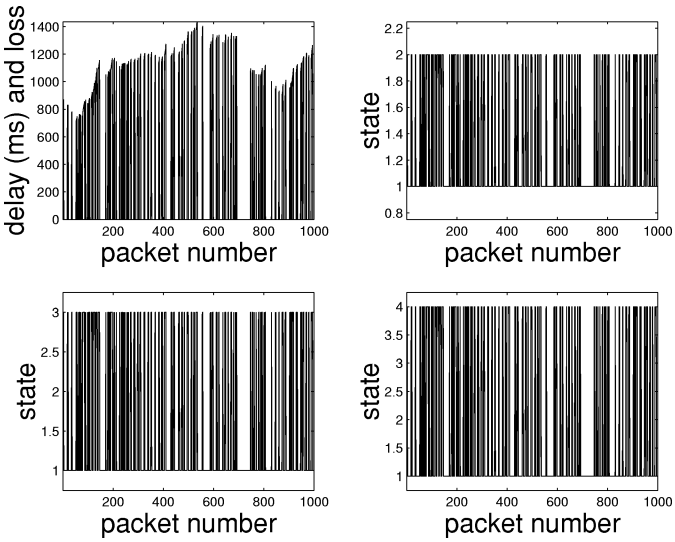

Fig. 13. State-sequence estimation before learning (sequence $n.\ 21$, Example I): test sequence and two-, three-, and four-state starting models, respectively.

The log-likelihood was evaluated for every test sequence. Circles and asterisks in Fig. 10 correspond to the log-likelihood for test sequences evaluated, respectively, for starting and trained models. It can be noted how the trained models exhibit an almost constant log-likelihood, showing how the channel can be considered to have stationary statistical characteristics for that time interval. Meanwhile, the starting models exhibit lower and more variable values for log-likelihood.

Moreover, Figs. 11 and 12 show the hidden states of the starting and trained models on a test sequence, the sequence $n.\ 6$. Comparing state sequences from starting and trained models, it can be noted how they assume a very different behavior. In case of starting models, hidden states are strictly dependent from instantaneous behavior of the channel, showing a rapidly oscillating trend, while hidden states for the trained models seem to capture well the state of the network on a larger time scale, exhibiting a more stable trend.

All this confirms that a state is associated to a particular congestion level of the channel, characterized by its own loss probability (depending on parameters $\mathbf{p}$), by its own mean delay (depending on parameters $\{\gamma, \vartheta\}$), by its own duration in the state itself (depending on parameters $\mathbf{A}$), and by its own transition probabilities into other states (depending on parameters $\mathbf{A}$).

Figs. 13 and 14 refer to the sequence $n.\ 21$ where the channel statistics have changed, as suggested by Fig. 10, and the parameters of the model are not the best estimate. However, the trained models still seem to furnish some information with respect to the starting models.

3) Generative Model: The trained models have also been used as generators in order to simulate channel behavior in terms of packet losses and delays. They seem to represent well channel statistics, in term of throughput, autocorrelation, average loss, and average delay. The throughput (η) of the model was considered as the probability that a packet is delivered within the maximum allowed delay $(\tau_{\max})$

$$\eta(\tau_{\max}) = \sum_{i=1}^{N} \pi_i p_i \int_0^{\tau_{\max}} f_i(t)dt = \sum_{i=1}^{N} \pi_i p_i \Gamma\left(\frac{\tau_{\max}}{\vartheta_i}, \gamma_i\right).$$

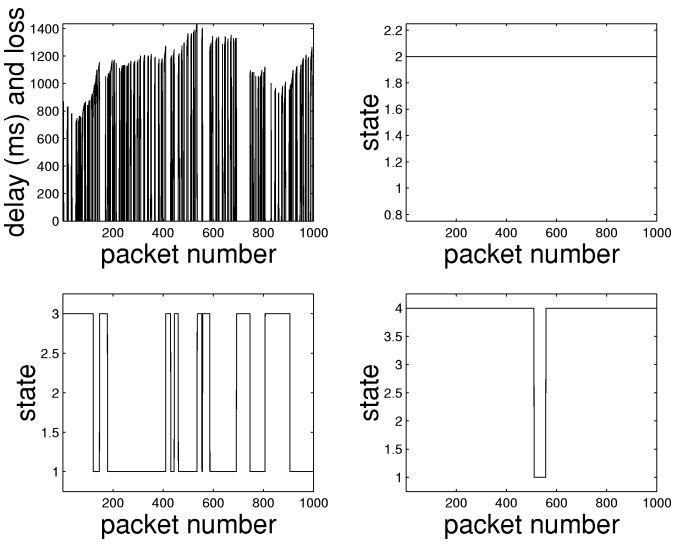

Fig. 14. State-sequence estimation after learning (sequence $n.\ 21$, Example I): test sequence and two-, three-, and four-state trained models, respectively.

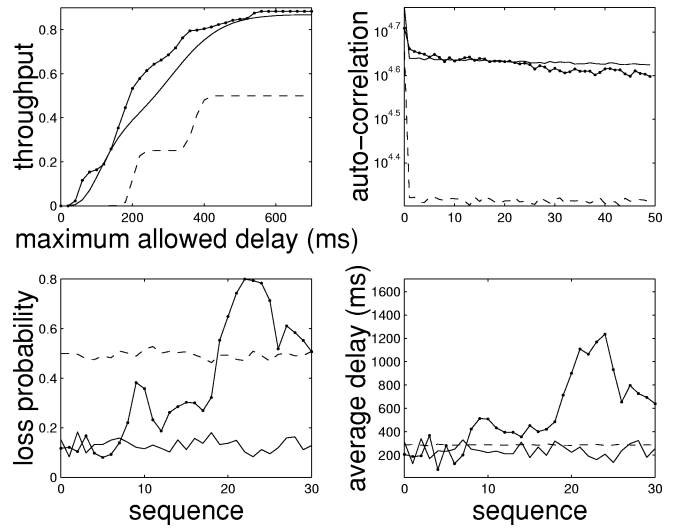

Fig. 15. Synthesis model capability in terms of throughput, correlation, loss and delay statistics, respectively (Example I), for two-state starting (dashed line) and trained (solid line) models with respect to real data (point–solid line).

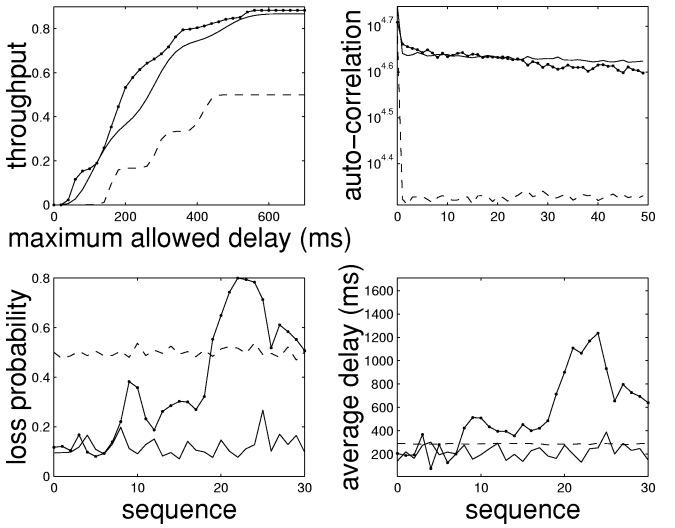

Fig. 16. Synthesis model capability in terms of throughput, correlation, loss and delay statistics, respectively (Example I), for three-state starting (dashed line) and trained (solid line) models with respect to real data (point–solid line).

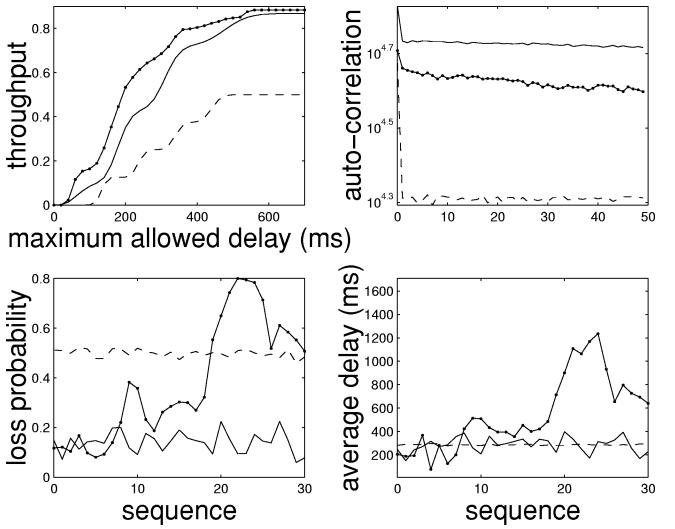

Fig. 17. Synthesis model capability in terms of throughput, correlation, loss and delay statistics, respectively (Example I), for four-state starting (dashed line) and trained (solid line) models with respect to real data (point–solid line).

The starting and the trained models were used to generate 31 000 outputs, via Monte Carlo simulations, which were grouped in blocks of $K = 1000$ consecutive samples. Each block represents a synthetic sequence. Auto correlation, loss probability, and average delay of the training sequence and of synthetic sequences generated by the starting and the trained models were evaluated. The autocorrelation $r_y(m)$ of the sequence $\mathbf{y} = (y_1, y_2, \ldots, y_K)$ was evaluated by use of the following formula:

$$r_y(m) = \frac{1}{K-m} \sum_{k=1}^{K-m} y_n y_{n+m}$$

and only $K_r = 50$ samples of $r_y(m)$ are considered, as for $m \to K$ the formula gives poor information because of the decreasing number of used sequence samples. Figs. 15–17 show the throughput, the autocorrelation, the loss probability, and the average delay for two-, three-, and four-state starting and trained models compared with real data.

B. Example II

The same steps of Section IV-A are briefly described in the following, with reference to measures performed in April 2003 between Università Campus Bio-Medico di Roma and Dipartimento di Ingegneria dell'Informazione, Seconda Università di Napoli, in which the interdeparture time is $T = 10$ ms, and the packet size is $N_b = 4000$ b ($R = 0.4$ Mb/s). The Internet path was composed of eight wired links (obtained by use of `traceroute`). The path has a bottleneck link of 0.5 Mb/s (nominal bandwidth).

Figs. 18–24 and Tables V–VIII refer to this example. Also in this case, the convergence during the learning algorithm is reached in a few iterations (see Fig. 18). The histogram of the training sequence shows a clear bimodal behavior that makes

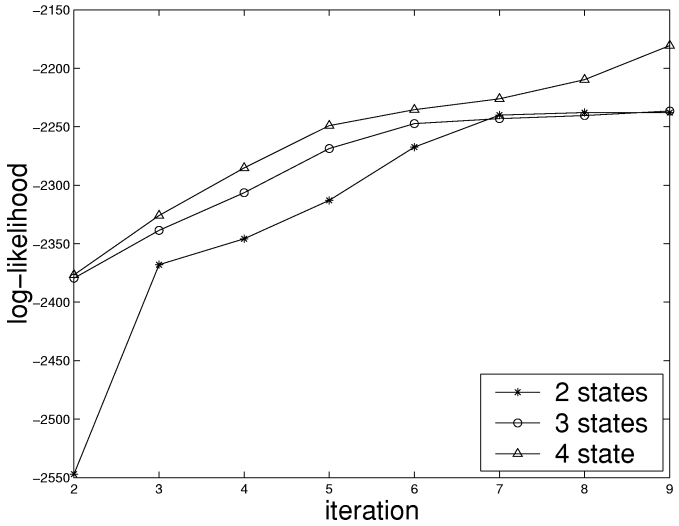

Fig. 18. Log-likelihood trend in the learning procedure (Example II).

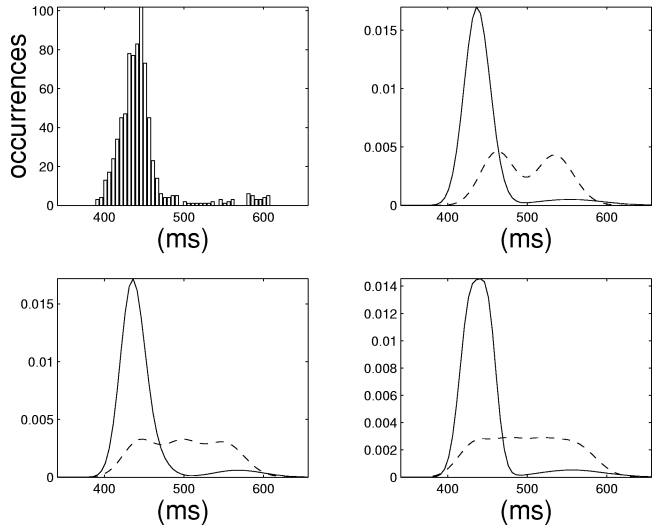

Fig. 19. Learning delays statistics (Example II): histogram of measured delays (the bin size is 5 ms) and continues term of pdf of two-, three-, and four-state starting (dashed line) and trained (solid line) models, respectively.

three-state and four-state models very close to the two-state model (see Fig. 19).

$$\begin{cases} \pi_{\mathrm{a}}^{(2)} = \pi_1^{(2)} = 0.930 \\ \pi_{\mathrm{a}}^{(3)} = \pi_1^{(3)} + \pi_2^{(3)} = 0.936 \\ \pi_{\mathrm{a}}^{(4)} = \pi_1^{(4)} + \pi_2^{(4)} + \pi_3^{(4)} = 0.929 \end{cases}$$

$$\begin{cases} \pi_{\mathrm{b}}^{(2)} = \pi_2^{(2)} = 0.070 \\ \pi_{\mathrm{b}}^{(3)} = \pi_3^{(3)} = 0.064 \\ \pi_{\mathrm{b}}^{(4)} = \pi_4^{(4)} = 0.071 \end{cases}$$

$$\begin{cases} P_{\mathrm{loss,a}}^{(2)} = \pi_1^{(2)} \left(1 - p_1^{(2)}\right) = 0.233 \\ P_{\mathrm{loss,a}}^{(3)} = \pi_1^{(3)} \left(1 - p_1^{(3)}\right) + \pi_2^{(3)} \left(1 - p_2^{(3)}\right) = 0.233 \\ P_{\mathrm{loss,a}}^{(4)} = \pi_1^{(4)} \left(1 - p_1^{(4)}\right) + \pi_2^{(4)} \left(1 - p_2^{(4)}\right) \\ \qquad + \pi_3^{(4)} \left(1 - p_3^{(4)}\right) = 0.232 \end{cases}$$

$$\begin{cases} P_{\mathrm{loss,b}}^{(2)} = \pi_2^{(2)} \left(1 - p_2^{(2)}\right) = 0.017 \\ P_{\mathrm{loss,b}}^{(3)} = \pi_3^{(3)} \left(1 - p_3^{(3)}\right) = 0.017 \\ P_{\mathrm{loss,b}}^{(4)} = \pi_4^{(4)} \left(1 - p_4^{(4)}\right) = 0.017 \end{cases}$$

$$\begin{cases} D_{\mathrm{mean,a}}^{(2)} = d_1^{(2)} = 437.55 \text{ ms} \\ D_{\mathrm{mean,a}}^{(3)} = \frac{\pi_1^{(3)}}{\pi_1^{(3)} + \pi_2^{(3)}} d_1^{(3)} + \frac{\pi_2^{(3)}}{\pi_1^{(3)} + \pi_2^{(3)}} d_2^{(3)} = 439.36 \text{ ms} \\ D_{\mathrm{mean,a}}^{(4)} = \frac{\pi_1^{(4)}}{\pi_1^{(4)} + \pi_2^{(4)} + \pi_3^{(4)}} d_1^{(4)} + \frac{\pi_2^{(4)}}{\pi_1^{(4)} + \pi_2^{(4)} + \pi_3^{(4)}} d_2^{(4)} \\ \qquad + \frac{\pi_3^{(4)}}{\pi_1^{(4)} + \pi_2^{(4)} + \pi_3^{(4)}} d_3^{(4)} = 436.65 \text{ ms} \end{cases}$$

$$\begin{cases} D_{\mathrm{mean,b}}^{(2)} = d_2^{(2)} = 556.44 \text{ ms} \\ D_{\mathrm{mean,b}}^{(3)} = d_3^{(3)} = 570.25 \text{ ms} \\ D_{\mathrm{mean,b}}^{(4)} = d_4^{(4)} = 558.81 \text{ ms}. \end{cases}$$

It can be noted that increasing the number of states of the model do not take any real advantage (see also previous equations). Because of the bimodal behavior of the channel, models with more than two states use their states to capture some transient behavior (see also Tables VII and VIII) from the principal two situations that have no practical relevance. This can also be noted via state-sequence estimation. Fig. 20 refers to state

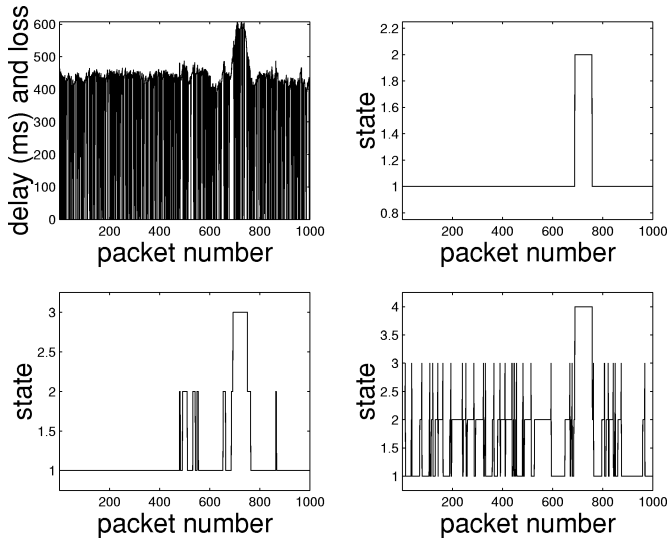

Fig. 20. State-sequence estimation (Example II): training sequence and two-, three-, and four-state trained models, respectively.

estimation for the training sequence, while Fig. 22 to state estimation for sequence $n.$ 3. Both of them show two different local behaviors of the channel. Fig. 21 shows the log-likelihood for test sequences evaluated for starting and trained models, while Fig. 23 shows the statistics of the starting and trained two-state models when used as channel simulators. Also in this case, the parameters of the trained models hold for sufficient time, i.e., the HMM-based strategy confirms its generalization capability. Fig. 24 shows the relations among the states of different trained models.

On the basis of the two examples previously described and other experiments that have been performed, the HMM-based modeling appears to be a good strategy as it is able to capture both loss and delay characteristics quite well.

Stable behavior of the states of a trained model suggests investigating on the possibility of supporting adaptive services mechanisms. Such online modeling features can be exploited

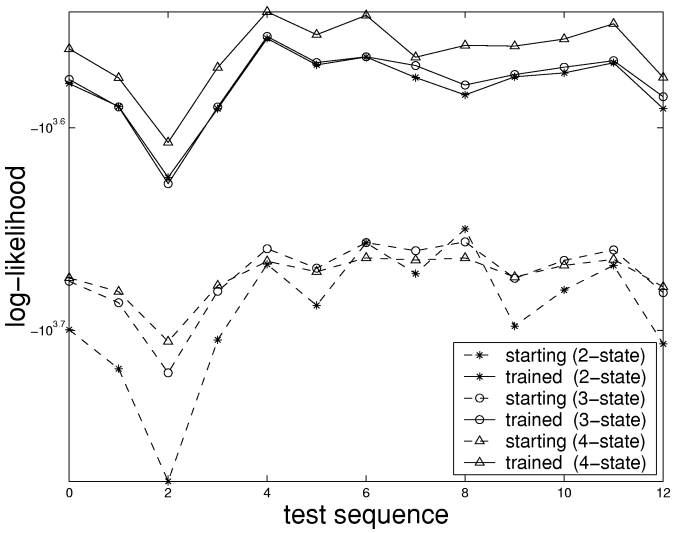

Fig. 21. Capacity of generalization of the model by means of the log-likelihood (Example II).

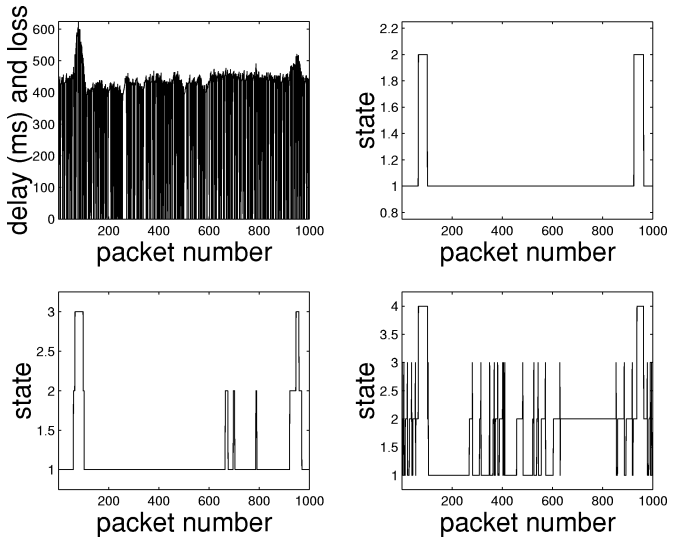

Fig. 22. State-sequence estimation after learning (sequence $n.\ 3$, Example II): test sequence and two-, three-, and four-state models, respectively.

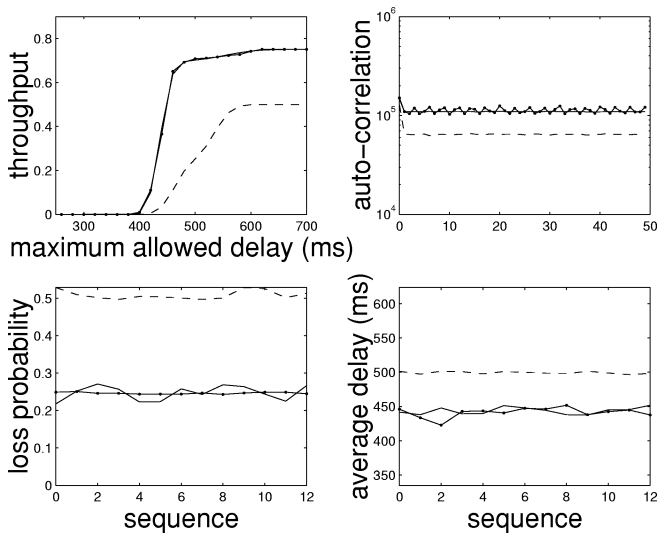

Fig. 23. Synthesis model capability in terms of throughput, correlation, loss and delay statistics, respectively (Example II), for two-state starting (dashed line) and trained (solid line) models with respect to real date (point–solid line).

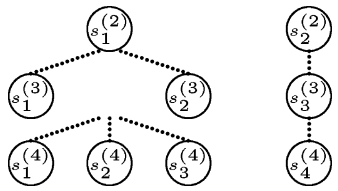

Fig. 24. Correspondences among states of the trained models (Example II).

to support device independent services as defined by the corresponding W3C working group,[2] according to the scheme shown in Fig. 25. Such a scheme needs losses and delays to be monitored to train an HMM, like the one previously described. Then state-sequence estimation could be used to foresee the short-term future behavior of the channel. This information could be sent back to the sender in order to adapt transmission. This strategy would clearly require that sufficient stationarity of the channel exists to make adaptive coding strategies worth the effort. We are working on optimum design for distributing information over packets [21], [25], we are working on the extension of the model to the case of nonperiodic UDP traffic [23], and we are working on adaptive transmission bit rate [24]; this would require that the estimated model holds for (slowly) variable interdeparture time or, better, would require an extension of the model to a more general scheme with a variable interdeparture time source. We believe this is possible in many practical situations, and we are currently pursuing such an effort.

When adaptive coding is not possible, or not worth the effort, good channel modeling can be very useful to evaluate performance of existing coders.

Other open problems are to determine how much a training sequence can be reduced, taking into account the tradeoff between the complexity and the frequency of the training step, as well as the proposal of a threshold algorithm evaluating when the training step has to be run again.

V. Conclusion

In this paper, we have proposed a Bayesian network with the objective of modeling end-to-end packet channel behavior, jointly capturing loss and delay characteristics. The proposed model generalizes the HMM description of real channels introducing a joint stochastic modeling of losses and delays. A training procedure, based on the EM algorithm, to learn parameters of the equivalent HMM was derived. Preliminary results are very encouraging, as the HMM is able to capture well loss and delay characteristics of the network. Tests run on real packets links showed how trained models exhibit good generalization capabilities. We also discussed the significance of hidden states automatically found by the training algorithm, showing how they can be associated to particular congestion levels of the network. Monitoring or even prediction of hidden

[2]W3C—Device Independence Working Group. [Online]. Avaliable: http://www.w3c.org/2001/di/

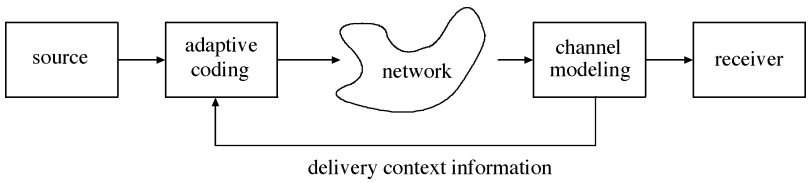

Fig. 25. Scheme for an adaptive communication protocol.

TABLE V
AVERAGE AND TRAINING STATISTICS COMPARISON (EXAMPLE II)

	training	starting	2-state	3-state	4-state
P_{loss}	0.249	0.500	0.249	0.249	0.249
D_{mean} (ms)	445.99	499.66	445.85	447.70	445.30

TABLE VI
STATE-CONDITIONED STATISTICS FOR A TWO-STATE MODEL (EXAMPLE II)

	π_1 $1-p_1$ d_1 (ms)	π_2 $1-p_2$ d_2 (ms)
starting	0.500 0.500 463.51	0.500 0.500 535.80
trained	0.930 0.250 437.55	0.070 0.237 556.44

TABLE VII
STATE-CONDITIONED STATISTICS FOR A THREE-STATE MODEL (EXAMPLE II)

	π_1 $1-p_1$ d_1 (ms)	π_2 $1-p_2$ d_2 (ms)	π_3 $1-p_3$ d_3 (ms)
starting	0.333 0.500 445.44	0.333 0.500 499.66	0.333 0.500 553.86
trained	0.811 0.213 435.65	0.125 0.478 463.28	0.064 0.259 570.25

TABLE VIII
STATE-CONDITIONED STATISTICS FOR A FOUR-STATE MODEL (EXAMPLE II)

	π_1 $1-p_1$ d_1 (ms)	π_2 $1-p_2$ d_2 (ms)	π_3 $1-p_3$ d_3 (ms)	π_4 $1-p_4$ d_4 (ms)
starting	0.250 0.500 434.59	0.250 0.500 477.97	0.250 0.500 521.34	0.250 0.500 564.72
trained	0.517 0.151 429.34	0.318 0.191 451.93	0.094 0.996 425.18	0.071 0.241 558.81

states should be very effective in the implementation of content-adaptive communication strategies. Future works will be directed toward model improvements and development of content-adaptive strategies based on hidden-state knowledge. Other interesting aspects are modifications of the model to verify its usefulness in a TCP scenario where states could be used to distinguish different typologies of losses (errors, congestion, etc.), as well as to take into account for different metrics such as interloss distribution and delay variation.

ACKNOWLEDGMENT

The authors would like to thank the Associate Editor and the anonymous reviewers for dedicating their time to the final version of this manuscript.

REFERENCES

[1] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell Syst. Tech. J.*, vol. 39, pp. 1253–1265, Sep. 1960.
[2] E. O. Elliott, "Estimates of error-rate for codes on burst-noise channels," *Bell Syst. Tech. J.*, vol. 42, pp. 1977–1997, Sep. 1963.
[3] L. A. Liporace, "Maximum likelihood estimation for multivariate observations of Markov sources," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 5, pp. 729–734, Sep. 1982.
[4] B. H. Juang, S. E. Levinson, and M. M. Sondhi, "Maximum likelihood estimation for multivariate mixture observations of Markov chains," *IEEE Trans. Inf. Theory*, vol. IT-32, no. 2, pp. 307–309, Mar. 1986.
[5] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–285, Feb. 1989.
[6] W. Turin and M. M. Sondhi, "Modeling error sources in digital channels," *IEEE J. Sel. Areas Commun.*, vol. 11, no. 3, pp. 340–347, Apr. 1993.
[7] J. C. Bolot, "Characterizing end-to-end packet delay and loss in the internet," *J. High-Speed Netw.*, vol. 2, no. 3, pp. 305–323, Dec. 1993.
[8] M. Zorzi, R. R. Rao, and L. B. Milstein, "On the accuracy of a first-order Markov model for data block transmission on fading channels," in *Proc. IEEE Int. Conf. Personal Communications*, Nov. 1995, pp. 211–215.
[9] J. A. Bilmes, "A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models," University of Berkeley, CA, Tech. Rep. ICSI-TR-97-021, 1998.
[10] W. Turin and R. van Nobelen, "Hidden Markov modeling of flat fading channels," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 9, pp. 1809–1817, Dec. 1998.
[11] V. Paxson, "End-to-end internet packet dynamics," *IEEE Trans. Netw.*, vol. 7, no. 3, pp. 277–292, Jun. 1999.
[12] W. Jiang and H. Schulzrinne, "Modeling of packet loss and delay and their effect on real-time mulrimedia service quality," presented at the Int. Workshop Network Operating System Support for Digital Audio Video, Chapel Hill, NC, Jun. 26–28, 2000.
[13] K. Salamatian and S. Vaton, "Hidden Markov modeling for network communication channels," in *Proc. ACM Sigmetrics/Performance*, vol. 29, 2001, pp. 92–101.
[14] V. K. Goyal and J. Kovacevic, "Generalized multiple description coding with correlating transforms," *IEEE Trans. Inf. Theory*, vol. 47, no. 6, pp. 2199–2224, Sep. 2001.
[15] J. Liu, I. Matta, and M. Crovella, "End-to-end inference of loss nature in a hybrid wired/wireless environment," presented at the Modeling Optimization in Mobile, Ad Hoc, Wireless Networks, Sophia Antipolis, France, Mar. 3–5, 2003.
[16] C. J. Bovy, H. T. Mertodimedjo, G. Hooghiemstra, H. Uijterwaal, and P. Van Mieghem, "Analysis of end-to-end delay measurements in internet," in *Proc. Passive Active Measurement Workshop*, Mar. 2002, pp. 26–33.
[17] S. Avallone, M. D'Arienzo, M. Esposito, A. Pescapé, S. P. Romano, and G. Ventre, "Mtools," *IEEE Network (*Networking Column*)*, vol. 16, no. 5, p. 3, Sep./Oct. 2002.
[18] P. Salvo Rossi, G. Romano, F. Palmieri, and G. Iannello, "Bayesian modeling for packet channels," in *Proc. Italian Workshop Neural Nets*, vol. 2859/2003, Jun. 2003, pp. 218–225.
[19] ——, "A hidden Markov model for internet channels," in *Proc. IEEE Int. Symp. Signal Processing Information Technology*, Dec. 2003, pp. 50–53.
[20] S. Avallone, A. Pescapé, and G. Ventre, "Analysis and experimentation of internet traffic generator," presented at the Int. Conf. Next Generation Teletraffic Wired/Wireless Advanced Networking, St. Petersburg, Russia, Feb. 2–6, 2004.
[21] P. Salvo Rossi, G. Romano, F. Palmieri, and G. Iannello, "Interleaving for Packet Channels," in *Proc. Conf. Information Sciences Systems*, Mar. 2004, pp. 1560–1564.
[22] P. Salvo Rossi, F. Palmieri, and G. Iannello, "HMM-based monitoring of end-to-end packet channels," in *Proc. IEEE Int. Conf. High Speed Network Multimedia Communications*, vol. 3079/2004, Jun. 2004, pp. 144–154.

[23] P. Salvo Rossi, A. P. Petropulu, J. Yu, F. Palmieri, and G. Iannello, "Internet loss-delay modeling by use of input/output hidden Markov models," in *Proc. IEEE Int. Workshop Multimedia Signal Processing*, Sep. 2004, pp. 470–473.

[24] S. Avallone, P. Salvo Rossi, V. La Marca, G. Iannello, F. Palmieri, and G. Ventre, "Congestion control for UDP Traffic," in *Proc. IEEE Int. Symp. Signal Processing Information Technology*, Dec. 2004, pp. 258–261.

[25] P. Salvo Rossi, F. Palmieri, G. Iannello, and A. P. Petropulu, "Packet Interleaving over Lossy Channels," in *Proc. IEEE Int. Symp. Signal Processing Information Technology*, Dec. 2004, pp. 476–479.

Pierluigi Salvo Rossi was born in Naples, Italy, on April 26, 1977. He received the "Laurea" degree in telecommunications engineering (*summa cum laude*) from University of Naples "Federico II," Italy, in January 2002 with a thesis on speech prosody modeling. He is currently working toward the Ph.D. degree in computer engineering at the Department of Computer Science and Systems, University of Naples "Federico II" and collaborates with the Department of Information Engineering, Second University of Naples, Italy.

In 2002, he worked as a Scientific Collaborator at CIRASS (Interdepartmental Research Center for Signal Analysis and Synthesis), University of Naples "Federico II." In 2003, he worked as a Scientific Collaborator with the Department of Information Engineering, Second University of Naples, Italy. In 2004, he was Visiting Research Scholar at the Communications and Signal Processing Laboratory (CSPL), Electrical and Computer Engineering Department, Drexel University, Philadelphia, PA. His research interests are in the areas of speech processing, signal processing for communication networks, signal and system modeling, wireless communications.

Gianmarco Romano received the "Laurea" degree in electronic engineering from the University of Naples "Federico II," Italy, in January 2000. He is currently working toward the Ph.D. degree in electronic engineering at the Department of Information Engineering, Second University of Naples, Italy.

From 2000 to 2002, he was Researcher at CNIT National Laboratory of Multimedia Communications, Naples, Italy. In 2004, he was Visiting Scholar with the Electrical and Computer Engineering Department, University of Connecticut, Storrs. His research interests are in the areas of multimedia communications, signal processing, wireless communications.

Francesco Palmieri was born in S. Maria C.V. (CE), Italy, on August 28, 1956. He received the "Laurea" degree in electronic engineering (*summa cum laude*) from University of Naples "Federico II," Italy, in 1980 and the M.S. degree in applied sciences and the Ph.D. degree in electrical engineering from the University of Delaware, Newark, in 1985 and 1987, respectively.

In 1981, he served as a second Lieutenant in the Italian Army in fulfillment of draft duties. In 1982 and 1983, he was a designer of digital telephone systems with ITT, FACE SUD Selettronica, Salerno, Italy, and Bell Telephone Manufacturing Company, Antwerpen, Belgium. He was appointed Assistant Professor in Electrical and Systems Engineering, University of Connecticut, Storrs, in 1987, where he was awarded tenure and was promoted to Associate Professor in 1993. From 1993 to 2000, he was Associate Professor in electrical communications with the University of Naples "Federico II." He is currently Full Professor in telecommunications with the Department of Information Engineering, Second University of Naples, Italy. His research interests are in the areas of signal processing, electrical communications, information theory, and neural networks.

Dr. Palmieri was awarded a Fulbright Scholarship from the University of Delaware in 1983.

Giulio Iannello was born in Rome, Italy, on August 17, 1957. He received the Electronic Engineering degree (*summa cum laude*) from the Politecnico di Milano, Milan, Italy, in 1981 and the Ph.D. degree in computer science from the University of Naples, "Federico II," Italy, in 1987.

In 1987, he joined the Department of Information Engineering and Applied Mathematics of the University of Salerno, Salerno, Italy, where he was a Senior Researcher until October 1992. In November 1992, he joined the Department of Computer Science and Systems of the University of Naples "Federico II," where he was an Associate Professor until October 2000. From November 2000 until April 2004, he was Full Professor at the same Department. He is currently Full Professor at University Campus Bio-Medico of Rome, Italy. He participated in several research projects funded by the Ministry of University and by the National Research Council. He is currently responsible for a work package in the FIRB (Fondo per gli Investimenti della Ricerca di Base) Project on "Middleware for advanced services over large-scale, wired-wireless distributed systems (WEB-MINDS)." His primary research interests include design and analysis of parallel algorithms, performance evaluation of parallel and distributed systems, communication software for high-performance interconnection networks, network protocols for Quality of Service (QoS), wireless networks. He has published more than 50 journal and conference papers in these areas.

Dr. Iannello is a member of the IEEE Computer Society, the Association for Computing Machinery (ACM), and the Associazione Italiana per l'Informatica ad il Calcolo Automatico (AICA).