

Modeling of temporal dependence in packet loss using universal modeling concepts

Raghavendra Singh and Antonio Ortega

Abstract

Packet losses are commonplace over the Internet, and can severely affect the quality of delay sensitive multi-media applications. In the current Internet architecture it is up to the application to react to the perceived congestion level in the network. The ability of the application to react is enhanced by the availability of simple and efficient loss models. Traditionally, Markov chain models have been proposed for modeling of the dependency in packet loss. In this paper we establish the drawbacks of using a Markov chain model and instead propose the more general Markov tree model for modeling the temporal dependency. Markov tree models are an example of universal models. In universal modeling, the most appropriate model for the observed data is chosen from a collection of models. The description length of the data with respect to a model is used as the performance measure so that the model which gives the minimum description length for the data is chosen as the best model. The minimum description length principle reflects the complexity of the model and hence prevents over-fitting of the data by too complex models. Our results show the advantage of using the Markov tree model.

The first author is with the IBM India Research Lab., New Delhi, India. The second author is with the Integrated Media Systems Center and the Dept. of Electrical Engineering at University of Southern California. [raghavsi@in.ibm.com,ortega@sipi.usc.edu]. This work was done by the first author as part of his PhD. thesis when he was at IMSC. This research has been funded in part by the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, Cooperative Agreement No. EEC-9529152 and by the National Science Foundation under grant MIP-9804959.

I. INTRODUCTION

Increasing traffic on the network has led to packet losses and delay becoming commonplace over the Internet. These losses and delay can cause severe degradation in the quality of delay constrained multimedia applications such as audio/video conferencing and live streaming. With the current Internet architecture not offering any guarantee on quality of service it is upto the multimedia application to adjust to the perceived congestion level in the network and ensure a suitable end-user quality. The applications can do so by performing rate control, for example by adapting their encoding rate (bits per sample) [1], or by changing the number of layers they will send [2], and (or) performing error control by adjusting the relative allocation between the source coding and channel coding bits [3], [4]. For such adaptive applications it is important to have simple packet loss models that can be parameterized in an on-line manner [5]. These loss model allow the application to adopt, with minimal delay, the optimal strategy, e.g., the optimal transmission rate that would minimize the expected distortion at the decoder, given the present condition of the network.

A number of studies have shown that packet losses exhibit a finite dependence in time [5], i.e., the probability of the current packet being lost is dependent upon whether the past few packets have been received or lost. Specifically, if a lost packet is represented by the symbol one and a received packet by the symbol zero, then the packet loss process can be modeled as a finite memory binary random process, i.e., a binary Markov process. These processes can be modeled by a finite memory binary model, where the probabilities of the next symbol being one or zero are conditioned on a function of a finite number of contiguous past observations. An example of such a model is the binary Markov chain model [5]. In a Markov chain model of order k , the “context”, i.e., the function of the past observations, of a symbol is the string of past k symbols. Thus there are $N_C = 2^k$ contexts in this model each with its own conditional probability. An advantage of the Markov chain model is that a finite state machine (FSM) can be associated with it, where each state corresponds to a context and the associated conditional probabilities. Given the present state, and the status of the current data packet, lost or received, the machine can transit to the next state according to the FSM rules. Hence these models are easy to implement and are used in many multimedia schemes [1], [3].

However when fitting Markov chain models to the data by estimating the conditional probabilities of the contexts, a number of difficulties arise [6]. In a chain model if the order of a model is increased to find a better fit there is a corresponding exponential increase in the number of states. This increase in states will not only increase the complexity of the system but more importantly “over-fit” the data, adversely affecting the performance of the model. An example of over-fitting of data is “context dilution” [7] which occurs when the source statistics have to be spread over too many contexts leading to inaccuracy in the corresponding estimates.

Let us take a concrete example, the Markov chain model of order 3 for trace-27 (Section II) is shown in Table I. The counts in the table, one for each symbol, 0 and 1, are the number of time the symbol occurs for the given context, e.g., in the trace the number of times a packet is lost (symbol 1), given that the last three packets have been received (context 000), is 3700. These counts can be used to calculate the conditional probabilities for the given context and symbol,

Context	Counts (0,1)	Context	Counts (0,1)
000	360639, 3700	100	3702, 155
001	2989, 864	101	158, 46
010	2984, 163	110	872, 40
011	805, 106	111	106, 141

TABLE I

COUNTS FOR 3RD ORDER MARKOV CHAIN MODEL OF TRACE-27 (TABLE II).

for example the probability of the next packet being lost given that the last three packets have been received is about 0.01. From the table, the counts for contexts (110), (101), (011), (111) are very low. This implies that these contexts do not occur with any regularity and if they are used for modeling they may not provide enough information about the process. Removal of these states might actually help in improving the performance of the model. Another example of over-fitting in a Markov chain model is that some of the contexts could be equivalent because they have identical (or nearly identical) conditional probabilities. For example in the table the counts of contexts (100), (010) and (001) are very similar; lumping together of these equivalent contexts would reduce the number of parameters in the model without necessarily affecting the performance of the model.

From the above discussion one should wonder whether contexts (states) can be arbitrarily deleted or lumped. As shown by Weinberger *et al.* [6] this is not possible; an arbitrary removal of redundant parameters would most likely destroy the finite state machine property of the Markov chain and might lead to a model which is not Markovian in nature. For class of sources which have a finite dependence in time, Weinberger *et al.* have proposed an alternative universal finite memory model. This model which will be referred to as the Markov tree model was first developed by Rissanen [8]. A simplistic view of the model is that it is a collection of all possible Markov models, from which the most appropriate model is chosen for prediction of a symbol of the source [9]. Thus, rather than deleting or lumping contexts of the Markov chain from the set of all possible contexts the best set is chosen, i.e., from an “over-complete” model, a model which captures the essential information of the source is chosen.

The example given in Table I shows the shortcoming of the traditional Markov chain models for modeling of the packet loss process. This is further established with the help of experiments later in this paper. If there is a mismatch between the model used by the application and the actual network behavior, the performance of the application will suffer accordingly [1]. Thus there is a need for an alternative model for modeling the dependency in packet losses and for this purpose we propose to use universal modeling concepts. This is an area where to the best of our knowledge universal modeling has not been used before. The results in this paper show that the tree models avoid over-fitting of the packet loss data and hence give better modeling performance. Further, the non-stationary characteristics of the Internet network traffic [10] are more suitably modeled by these tree models. A Markov chain model can adapt to changing data

	Date	Type	Sampling Interval	Destination	Duration	Loss Rate
Trace-25	20Dec97	unicast	20ms	Seattle	2hrs 27min	1.7%
Trace-27	21Dec97	unicast	20ms	Los Angeles	2hrs	1.4%

TABLE II
TRACE DESCRIPTIONS FROM [5]

conditions by changing the probability distributions associated with each context. The Markov tree model goes one step further, it not only adapts the probability distribution but also the set of chosen contexts.

The paper starts with a review of related work in section II followed by a brief introduction to mathematical concepts and universal modeling in section III. In section IV we motivate the use of a tree model over a chain model with experiments results. Finally, we conclude with section V.

II. RELATED WORK

There has been a lot of work on measuring, analyzing and modeling network traffic [11], [12], [13]. We are primarily concerned with Yajnik *et al.*'s work because they have analyzed real network traces and modeled the temporal dependence of packet losses in these traces as Markov processes. Their main analysis method is based on the auto correlation lag present in the binary trace data. In their experiment, the lag is varied and if the corresponding correlation is greater than a statistical measure, then the data is considered correlated for that lag. The minimum lag beyond which the process can be considered independent is used to estimate the order k of a Markov chain model. Once the order has been chosen, the probability distributions can be adapted on the fly for the set of contexts associated with the model. Out of the 38 trace segments considered, the Bernoulli model was found to be accurate for 7 segments, the 2-state model was found to be accurate for 10 segments and for the rest of the traces higher models, up to the order of 40, were proposed.

In this paper we use the traces measured by Yajnik *et al.* We have analyzed two different sets of traces, each of which is about 2 hour long and has been found stationary by the method proposed in [5]. The descriptions of the traces are given in Table II.

In [14] the authors have used hidden Markov models (HMM) for modeling the temporal dependence in packet loss. Out of the 36 traces analyzed by the authors, including the ones provided by Yajnik *et al.*, 35 traces are modeled by HMMs having less than four states while one trace is modeled by a 4-state HMM. We have not compared against these models. We believe that the use of hidden states will alleviate the problem of context dilution, however this will come at the cost of the additional complexity of the expectation maximization algorithm [15] that is used to estimate parameters. Due to this complexity it is unclear whether the HMM can be used to adapt in realtime to the non-stationary characteristics of the network. On the other hand in the proposed Markov tree models the complexity of estimating the parameters (probability)

associated with a state is exactly the same as that of the Markov chain model. If the number of states in the chain model and the tree model are same, then the additional complexity in the tree model is due to the fact that unlike chain models to find the next state a tree search algorithm has to be used. By using the algorithm proposed by Furlan [9] this tree search can be implemented very efficiently.

III. PRELIMINARY MATHEMATICAL CONCEPTS AND INTRODUCTION TO UNIVERSAL MODELING

Let $\{x_1..x_n\}$, a string of binary random variables, be the information source we are considering. At each time instant i , after having scanned past data $x^i = x_1x_2..x_i$ the modeling problem is to make inferences on the next symbol x_{i+1} by assigning a conditional probability distribution $p(\cdot|c(x^i))$ to it, where context $c(x^i) \triangleq f(x_i..x_{i-k_i})$, $i - 1 \leq k_i \leq 0$. In the long run, the modeling goal is to maximize the probability

$$P(x^n) = \prod_{i=0}^{n-1} p(x_{i+1}|c(x^i)) \quad (1)$$

assigned to the entire sequence [7]. Taking the negative of the logarithm (base 2) of the above equation we see that the goal is to minimize $h(x)$,

$$h(x) = -\log_2(P(x^n)) = -\sum_{i=0}^{n-1} \log_2 p(x_{i+1}|c(x^i)), \quad (2)$$

where $h(x)$ is Shannon's entropy (code length) [16]. Given a model, entropy of the observed data will be small if the model's predictions match the observed data, else the entropy will be large. We use the measure of entropy to compare the performance of different models.

Clearly there are two steps to modeling the source. The first is to define the context $c(x^i)$ for a given x^i and x_{i+1} . The second is to estimate the conditional probability $p(x_{i+1} = a|c(x^i) = y_j)$, where $y_j \in \mathcal{S}$ and $a \in \mathcal{A}$. (\mathcal{S} is the context space and \mathcal{A} is the alphabet set). The probabilities can be estimated by storing the count, $n(a|y_j)$, i.e., the number of times symbol a occurs with context y_j . Each time x_{i+1} follows a given context $c(x^i)$, $n(x_{i+1}|c(x^i))$ is increased by one. The Laplacian estimate of the probabilities can then be defined as [9]

$$p(x_{i+1} = a|c(x^i) = y_j) = \frac{n(a|y_j) + 1}{\sum_{a \in \mathcal{A}} n(a|y_j) + 2}. \quad (3)$$

Other methods including those developed by Yajnik *et al.* could also be used for online probability estimation.

In the k th order Markov chain model used by Yajnik *et al.*, the context is defined as $c(x^i) = (x_i..x_{i-k})$, i.e., the memory is constant for all i and independent of x_{i+1} . Thus, once the order k has been defined the modeling problem reduces to estimating the probability distributions associated with each context. An important feature of the Markov chain model is that given the present context $c(x^i)$, the present symbol, x_{i+1} , the next context $c(x^{i+1})$ can be found

$$c(x^{i+1}) = g(c(x^i), x_{i+1}).$$

Hence a finite state machine can be associated with the model; this obviously reduces the implementation cost of the model. However there are limitations associated with a Markov chain model which have already been discussed in the introduction.

An alternate Markov model, the tree model, has often been used to model finite memory sources [6]. In a tree model of order k the set of all possible contexts $\{c(x^i) = (x_i \dots x_{i-k_i})\}$, where $k_i = 0 \dots k$, is defined. Given x^i , the goal is to estimate the “best” context $c_b(x^i)$ such that the probability $p(x_{i+1}|c_b(x^i))$ is minimized *and* the string associated with $c_b(x^i)$ is as short as possible. The second requirement reflects the complexity associated with the model itself and thus avoids over-fitting of the data. This process can be efficiently implemented by using the Context algorithm developed by Rissanen [8].

In the Context algorithm each context defines a node in a tree. This is the node which is reached by the path starting at root of the tree with the branch x_i followed by the branch x_{i-1} and so on. Thus for a k order tree model a k depth tree with $N_T = \sum_{i=0}^k 2^i$ number of nodes is defined. Each node stores the counts needed to estimate the probability using equation (3). The goal is to find the best subtree, where the leaves of the subtree define a subset of contexts, which minimizes the entropy of the observed data. The Context algorithm accomplishes this in two interleaved stages, the first stage grows a large tree and the second stage selects from the grown tree a particular context $c_b(x^i)$ for each $i > 0$. There are many versions of the Context algorithms, we have implemented the version by Furlan [9]. This algorithm uses a relative efficiency counter which compares the coding efficiency of a child node with the coding efficiency of the parent node. Coding efficiency is the entropy of coding a symbol given the probability distribution associated with a node. By using this relative efficiency counter the complexity of a full tree search is avoided; to find the best node (context) in a k depth tree at any instant i , k comparisons are needed to find the branch in the tree associated with x^i and at most $k - 1$ comparisons are needed to find the node along the branch which has the highest relative efficiency. In addition to be adaptive, $2k$ updates are needed to update the count and the relative efficiency counter for each of the node along the branch. This is implemented to provide realtime adaptation and prediction using a recursive tree structure.

IV. EXPERIMENTAL RESULTS

In the experiments given below we use the entropy of observed data, given a probability distribution, as the performance measure of the model. We are looking for the model for which the data has the lowest entropy. First we use Markov chain models of varying order to model the traces, trace-25 and trace-27. The order of the model is set and the conditional probability for each context in the model is estimated using equation (3). The entropy of the observed data is plotted versus the order of the model in Fig. 1. From the figure we notice that beyond an order of 5 there is no change in the entropy of the data. In fact for higher order models there is a deterioration in the performance of the model. This is due to context dilution, the higher order models have many states which do not have enough information about the data and thus are not making reliable estimates of the data. Clearly, if Markov chains were to be used to analyze or model the dependency of packet loss, high order Markov chains will not give good modeling

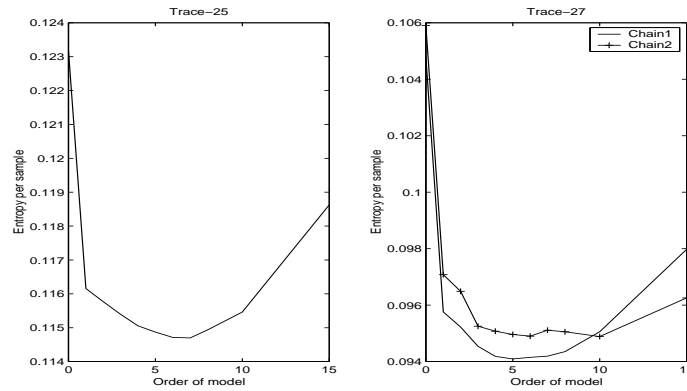


Fig. 1. Entropy results for Markov-chain modeling of the two traces. It can be seen that the minimum entropy is for chains of order at most 5, and there is no advantage in increasing the order. For the Chain1 experiment the counts $n()$ are initialized to zero and then adapted on the fly from trace-27. In Chain2 experiment the counts $n()$ are initialized by trace-25 and are then adapted when model is run for trace-27. Chain2 shows the adaptive nature of the chain model. The number of contexts in the model are 2^k if k is the order of the model.

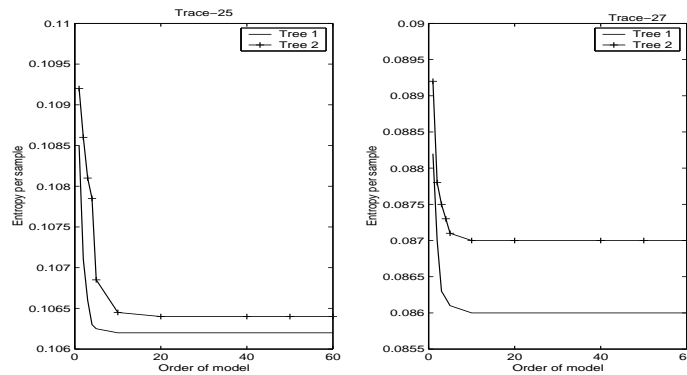


Fig. 2. Entropy results for the Markov tree modeling. Tree1 are the results when the counts $n()$ of the models are initialized to zero. For Tree2 the counts are initialized with one trace and the model is then used for modeling the other trace, e.g., for Tree2 results of trace-27, model is initialized by trace-25 and then run for trace-27. Tree2 shows the adaptivity of the model. The number of contexts in the model are $\sum_{i=0}^k 2^i$ where k is the order of the model.

performance.

In Fig. 2 we present entropy results for Markov tree models for trace-25 and trace-27. The Tree1 results are when the counts $n()$ are initialized to zero. We see that the minimum entropy for trace-25 occurs for a tree of order 35 while for trace-27 tree of order 25 is required. Further, notice that using these models the entropy of source reduces by at least 7% over the best Markov chain model performance, which implies that the tree model is a better model. Also from the Tree2 results we can see that the tree model adapts very well; for Tree2 the models are initialized with one trace and then run for the other trace. For example in the experiment for trace-27 the model is initialized by trace-25 and then run for trace-27, the result show that if the network conditions change drastically over a period of time the tree model will still perform very well.

In the Context algorithm the second stage of the algorithm selects a context, among the set of

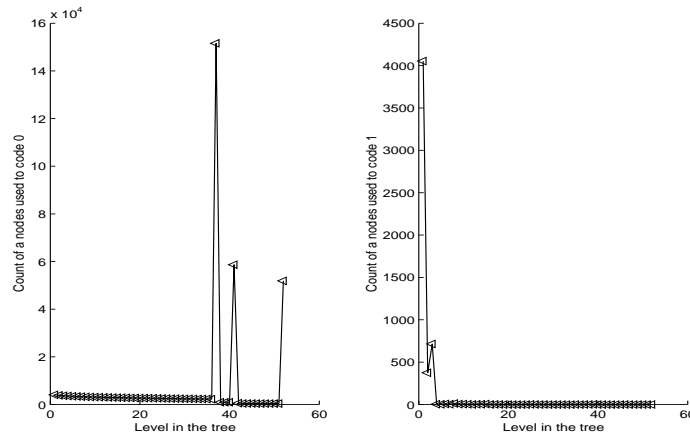


Fig. 3. The number of times a node (context) of the is used to code a symbol is analyzed. As the number of nodes are very large for this tree of depth 45, the counts of all the nodes on a level of the tree are accumulated and plotted accumulate their counts. The figure shows that the subtree is unbalanced.

contexts, to code the symbol. Thus the algorithm derives a subtree, i.e., a subset of best contexts, from the complete tree of the Markov tree model. If this sub-tree is balanced, i.e., all the leaves are of the same depth, then a Markov chain model can be used for modeling. In the experiments below we show that the derived subtree is highly unbalanced.

We calculate the number of times a node of the tree is used to code a symbol. As the number of nodes in a fully grown tree are very large, we group together the nodes in a level of the tree and accumulate their counts. In Fig.3 the counts for different levels of a tree of depth 45 are given. The tree is used for modeling the trace-27. The figure shows that there are nodes on level 32 and on level 42 of the tree which are used very frequently to code the symbol 0. However for coding symbol 1, nodes at level 2,3 and 4 are used. In other words, when coding symbol 1 a context of smaller memory is required than when coding symbol 0. Clearly, this is not possible in a Markov chain model.

Fig. 3 does not reveal whether the tree is unbalanced within a level, e.g., are all the nodes on level 32 used for coding symbol 0 or is there one node which is used more frequently than others? To answer this question, we again use the concept of entropy, the entropy of the probability of using nodes within a level is calculated. If all the nodes are being used equally the usage-entropy is going to be very close to one. If a small subset of nodes are used with more regularity than others, then the usage-entropy will be small, else the usage-entropy will be large. In Fig. 4 the usage-entropies for coding symbol 0 and 1 are plotted for various levels of a tree of depth 45 which is modeling the trace-27. The figure shows that for symbol 0 a very small subset of nodes is being used in each level. For symbol 1 the usage-entropy is low till level 4 after which it is very high, this is mainly because nodes on higher level are not being used regularly for coding symbol 1. Thus clearly the tree is not only unbalanced across but also within levels of the tree.

The above results clearly establish the superiority in performance of the Markov tree models for modeling of dependency in packet losses. However the advantages of the tree model come at the cost of additional states (nodes) and the associated complexity. To have fair comparison

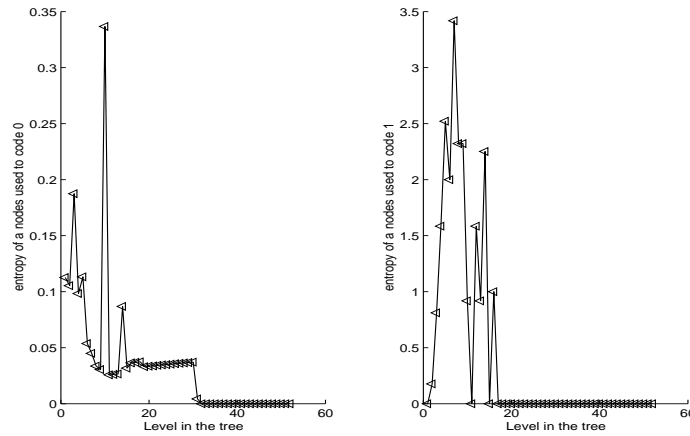


Fig. 4. The entropy of usage of nodes within a level is calculated using the probability of using a node to encode a symbol. If nodes are being equally used then the usage-entropy will be close to one. If only a very small subset of nodes in a level are being used then the usage-entropy will be very small else the usage-entropy will be high. The experiment establishes that tree is not only unbalanced across but also within levels.

both the tree model and the chain model need to have the same number of states. In the universal coding algorithm, as shown in the last two experiments, there is a subtree whose nodes are used more often than others because of their coding efficiency. A tree pruning algorithm can be used to explicitly choose these nodes and prune the other nodes. Then a variation of the Context algorithm, i.e., where the update of the coding efficiency of a node and the tree growing step is turned off, can be used to model the source. This model will be referred to as the pruned tree model, note that in this model the counts needed for estimating probability distributions are still estimated on the fly. The basic idea behind the model is fix the structure of the model based on some learning and then adapt the probability distributions. In Fig. 5 the results of coding the trace-27 with the pruned tree model are given along with the results of coding with a Markov chain model with similar number of states. For Pruned Tree-1, the Context algorithm is run on trace-27. From the full tree, the best set of nodes (contexts) is pruned and the corresponding counts set to zero. This set of nodes is used to then model trace-27. On the other hand, for Pruned Tree-2 Context algorithm is run on trace-25 and the best set of nodes is pruned from the tree and counts initialized to zero. The pruned tree is then used to model trace-27. For both the pruned tree experiments and the chain model experiment the counts $n()$ are initialized to zero and then learned on the fly. We can see from the results that though the pruned tree structure has been derived from a different trace in Pruned Tree-2, it still gives better results than an adaptive Markov chain model.

In the above results we are comparing the modeling performance; clearly the Markov tree model, models the temporal dependence in packet losses better than the Markov chain models. We are interested in using these models as predictors, such that the multimedia application can use the predicted future to react to the current network status and adopt an optimal strategy. The experiment that we are proposing compares the available rate predicted by a model and the actual rate calculated using the trace. Let the channel capacity be C , then the average actual rate

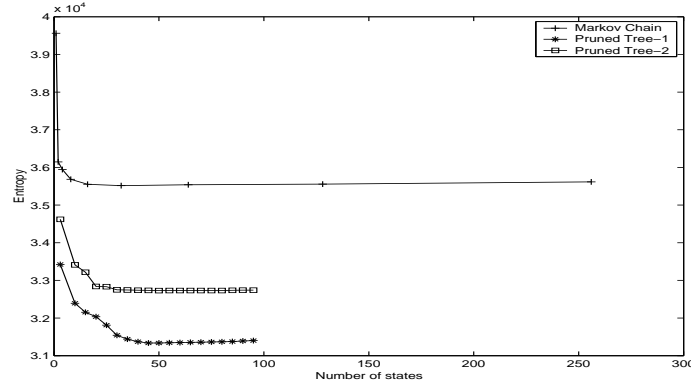


Fig. 5. Results comparing the pruned tree model with the chain model. Results are for trace-27. All results are adaptive, in that the probability models are learned on the fly. In Pruned Tree-1 the set of pruned nodes (best set of contexts) has been derived by running Context Algorithm on trace-27 and pruning the resultant tree. In Pruned Tree-2, the set of contexts has been derived from trace-25, i.e., Context Algorithm is run on trace-25 and the best set of contexts are extracted. Pruned Tree-1 performance is very close to the Markov tree model's performance. Pruned Tree-2 shows the adaptivity of the pruned tree model.

\bar{C} observed by a trace of length N is,

$$\bar{C} = \frac{\sum_{i=0}^{N-1} x_i * C}{N}$$

where x_i is the symbol zero or one in the trace. On the other hand, at any instant i given the context $c(x^i)$ if the probability of the next packet being received at the receiver is $p(x_i = 0|c(x^i))$ then the average available rate \tilde{C} is,

$$\tilde{C} = \sum_{i=k}^{N-1} p(x_i|c(x_i)) * C.$$

Given the trace $c(x^i)$ and the $p(x_i = 0|c(x^i))$ can be found for both the chain and the tree model. In Table III we are comparing the average available rates for a 8th order chain model with a 8th order tree model. In Fig. 6 we plot x_i from Trace-25 for a randomly chosen interval. $x_i = 0$ means that the packet has been received, from the figure packets at time instant 11 and 16 are lost. In the figure, for each of the models, the probability of a packet being received is also shown. For the chain model the order of the context is fixed at eight hence it is slow to react from the first packet loss. On the other hand for the tree model though the order of the model is also eight, the context it is using after the second packet loss is of a smaller order, hence it can react better to the packet losses.

V. CONCLUSION

In this paper we have established that Markov tree models, i.e., models based on the universal modeling concept are suitable for modeling of dependency in packet loss and have some preliminary results for showing how modeling of the network conditions affect a multimedia

	Trace-25	Trace-27
\bar{C}	0.9831*C	0.9865*C
\hat{C} Chain Model	0.9807*C	0.9831*C
\tilde{C} Tree Model	0.9825*C	0.9850*C

TABLE III

RESULTS COMPARING AVERAGE ACTUAL RATE OF A TRACE \bar{C} WITH THE EXPECTED AVAILABLE RATE OF A MODEL OF THE TRACE \tilde{C} .

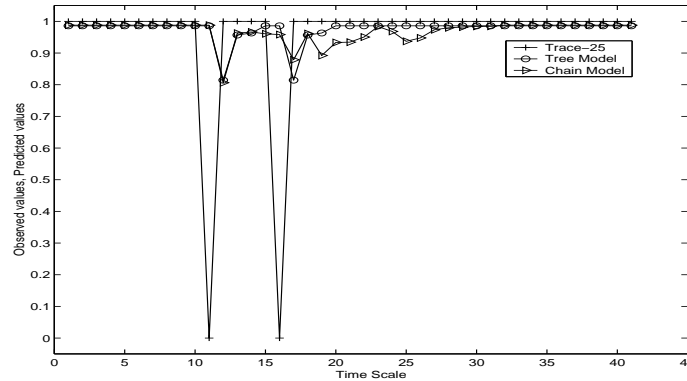


Fig. 6. A randomly selected interval of trace-25 is shown in the figure. A value of zero for Trace-25 implies that the packet has been received while a value of one implies that it has been lost. The probability of a packet being received at each time instant is also shown for both the models.

applications performance. This is an area we are currently exploring and we propose to make it a part of future work.

The tree models can be used for predicting the network behavior exactly like the chain models except that the tree models do not have FSMs associated with them. Instead the transition rules as described in the Context algorithm have to be used to find the next context (node) given the present and past symbols. As part of future work, we want to fit a FSM to the pruned tree model so as to make its complexity equivalent to the chain model. A FSM can be associated with any Markov tree model, however in this FSM there will be a large number of states but a small number of parameters, i.e., not each state will have a unique context and probability distribution [8]. However to the best of our knowledge there is no method for associating a minimal FSM, i.e., one with the minimum number of states, to the tree model, this is another area we are looking at.

REFERENCES

- [1] C.-Y. Hsu, A. Ortega, and M. Khansari, "Rate control for robust video transmission over burst-error wireless channels," *IEEE Journal on Selected Areas in Communications, Special Issue on Multimedia Network Radios*, vol. 17, pp. 756–773, May 1999.
- [2] Z. Miao and A. Ortega, "Optimal scheduling for streaming of scalable media," in *Proc. of 34th Asilomar Conference on Signals and Systems*, 2000.
- [3] J. Bolot and T. Turletti, "Adaptive error control for packet video in the internet," in *Proc. of ICIP*, 1996.

- [4] W. Jiang and A. Ortega, "Multiple description coding via polyphase transform and selective quantization," in *Proc. of VCIP (SPIE)*, 1999.
- [5] M. Yajnik, S. Moon, J. Kursoe, and D. Towsley, "Measurement and modelling of the temporal dependence in packet loss," in *Proc. of INFOCOM*, 1999.
- [6] M.J. Weinberger, J.J. Rissanen, and M. Feder, "A universal finite memory source," *IEEE Trans. on Information theory*, pp. 643–652, 1995.
- [7] M.J. Weinberger, J.J. Rissanen, and R. B. Arps, "Applications of universal context modeling to lossless compression of gray-scale images," *IEEE Trans. on Image Processing*, vol. 5, Apr. 1996.
- [8] J. Rissanen, "A universal data compression system," *IEEE Trans. on Information theory*, pp. 656–664, 1983.
- [9] G. Furlan, "An enhancement to universal modeling algorithm context for real time applications to image compression," in *Proc. of ICAASP*, 1991.
- [10] Y. Zhang, V. Paxson, and S. Shenker, "The stationarity of internet path properties," ACIRI Technical Report, May 2001.
- [11] V. Paxson, "End to end internet packet dynamics," *IEEE Trans. on Networking*, pp. 277–292, 1999.
- [12] J. Bolot, "End-to-end packet delay and loss behaviour in the internet," in *Proc. of SIGCOMM*, 1993, pp. 289–298.
- [13] L. Perret, "Simulation and modeling of internet packet losses with applications to video transmission," Travail de Diplome, Section Electricite, EPFL, Lausanne, Suisse, 1999.
- [14] K. Salamatian and S. Vaton, "Hidden markov modeling for network communication channels," in *Proc. of SIGMETRICS*, 2001.
- [15] L. R. Rabiner, "A tutorial on hidden markov models and selected application in speech recognition," *Proc. of IEEE*, vol. 77(2), pp. 257–285, Feb. 1989.
- [16] C.E. Shannon, "A mathematical theory of communication," *Bell System Tech. Journal*, vol. 27, pp. 379–423, 1948.