Failure Recovery Redundancy Overall Requirement

Emin Gabrielyan
EPFL and Switzernet Sàrl
Lausanne, Switzerland
emin.gabrielyan@epfl.ch

***Abstract* - Forward Error Correction (FEC) is very efficient for off-line streaming with large buffering time, but real-time streaming puts hard restrictions on the buffer size making FEC inefficient for combating long link failures on single path routes. Multi-path routing, orthogonal to buffering, can however make FEC effective also for real-time streaming. For this purpose we introduce a capillary routing algorithm offering layer by layer a wide range of multi-path routing topologies starting from simple solutions and evolving toward reliable and secure routing schemes with highly developed path diversity. The friendliness of a particular multi-path routing is rated by the overall amount of FEC redundancy required for combating the non-simultaneous failures of all links in the multi-path route. We rated the friendliness of a dozen of capillary routing layers, built on several hundreds of network samples obtained from a random walk wireless Mobile Ad-Hoc Network (MANET). The overall requirement in redundant FEC codes decreases substantially as the spreading of the routing grows.**

I. Introduction

Packetized IP communications behave like erasure channels. Data is chopped into packets, and since each packet is either received without error or not received, erasure resilient FEC codes can mitigate packet losses.

In off-line packetized applications Forward Error Correction (FEC) dramatically improves the quality and performance of communications under challenging network conditions [1]. Via satellite broadcast channel using erasure resilient Raptor codes [2] it is possible to simultaneously update voluminous GPS maps of millions of motor vehicles under conditions of arbitrary fragmental visibility and without a feed-back channel. In the film industry, LT codes [3] enable a fast delivery over the lossy internet of the day's film footage from the location it has been shot to the studio that is many thousands of miles away. These examples of off-line streaming significantly benefit from FEC due to the fact that contrary to real-time streaming, the application is not obliged to deliver in time the "fresh" packets and the buffer size is not a concern. When buffer size is restricted, FEC can only mitigate short granular failures. Many studies reported weak or negligible improvements when applying FEC to real-time streaming [4], [5], [6] and [7].

Studies stressing the poor FEC efficiency always assume that the media stream follows a single path. Exploiting multi-path routing "replacing" the long buffering time can nevertheless make FEC also efficient for fault-tolerant real-time streaming. There is an emerging body of a literature addressing the path diversity for improving the efficiency of FEC [8], [9], [10] and [11], however the routing patterns in these studies are limited to either two (possibly correlated) paths or in the best case to a sequence of parallel and serial links.

In this paper we present a comparative study within a wide range of multi-path routing patterns. Single path routing, being considered as too hostile, is excluded from our comparisons. Steadily diversifying multi-path routing patterns are created with *capillary routing*, where the routing suggestions are proposed layer by layer and the path diversity develops as the layer number increases (sections III and IV).

For evaluating a multi-path routing suggestion, we rely on the amount of the adjustable FEC needed for combating failures of individual links. Adjustable FEC for real-time streaming was proposed by several authors [4], [5] and [6]. In two-way real-time media, the packet loss rate information is usually transmitted with the opposite flow using Real-time Transport Control Protocol (RTCP). The sender increases the FEC overhead whenever the packet loss rate is about to exceed the tolerable limit. The friendliness of the underlying network routing is measured by Failure Recovery Redundancy Overall Requirement (FRROR), which is the total sum of all transmission rate overheads required from the sender during communication time. The novelty brought by FRROR is that a routing topology of any complexity can be rated by a single scalar value (section II).

In section 0, we evaluate the FEC efficiency of the capillarization by rating each layer of capillary routing with FRROR. Network samples are obtained from a wireless random walk Mobile Ad-hoc Network (MANET) with several hundreds of nodes. We show that multi-path routing, similarly to buffering, can substantially burst the efficiency of FEC and that the efficiency of FEC improves by developing the diversity factor of multi-path routing.

II. Redundancy Overall Requirement

Most real-time media streaming applications are tolerant to a certain level of packet losses due to passive error concealment or media encoding techniques. Voice over IP (VOIP) for example can tolerate 8% to 11% packet losses. The static tolerance can also be obtained or increased by a constant FEC code. We propose to combine the little static tolerance of the media stream, combating weak failures, with a dynamically added adaptive FEC combating the strong failures exceeding the tolerable packet loss rate.

For a given routing scheme FRROR is defined as the sum of all individual FEC transmission rate increases needed to combat each corresponding link failure. For example, if the communication footprint consists of five links, and in response to each individual link failure the sender increases the packet transmission rate by 25%, then FRROR will be equal to the sum of these five FEC transmission rate increases, i.e. to $5 \cdot 25\% = 1.25$.

Redundant packets (of approximately the same size as the media packets) are injected in the original media stream for every chunk of *M* source media packets using systematic erasure resilient codes (thus without affecting the original media packets). During streaming the number of media packets (*M*) in each chunk is supposed to stay constant. The number of redundant packets for each chunk of *M* media packets is however variable, depending on the conditions of the erasure channel. The *M* media packets with their related redundant packets form a FEC block. By $FEC_p \geq M$ we denote the FEC block size chosen by the sender in response to a packet loss rate *p*. We are assuming that the media stream has also a static tolerance to losses $0 \leq t < 1$ obtained with a constant FEC code, which by default streams the packets as FEC blocks of length of FEC_t. When the loss rate *p* measured at the receiver is about to exceed the tolerable limit *t*, the sender increases its transmission rate by injecting additional redundant packets.

The random packet loss rate, observed at the receiver during the time of a complete failure (or congestion) of a link in the communication path, is the portion of the traffic being routed toward the faulty link. Thus a complete failure of a link *l* carrying according to the routing pattern a relative traffic load of $0 \leq r(l) \leq 1$ will produce at the receiver a random packet loss rate equal to the same relative traffic load $r(l)$. The equation for FRROR can thus be written as follows:

$$FRROR = \sum_{l \in L \mid t \leq r(l) < 1} \left(\frac{FEC_{r(l)}}{FEC_t} - 1 \right) \qquad (1)$$

The links carrying the entire traffic are skipped in the sum index of equation (1), since the FEC required for the compensation of failures of such links is infinite. If for a given network topology the link is critical, any routing suggestion will unavoidably pass its entire traffic through that link, and therefore without affecting the comparison, the corresponding "equivalent" infinite components can be removed from the FRROR rates of all suggested routings. By construction (sections III and IV), none of considered multi-path routing schemes passes its entire traffic through a non-critical single link.

The FEC_p function we compute assuming a Maximum Distance Separable (MDS) code (e.g. a Reed-Solomon code). By the choice of an MDS code, the condition for a successful decoding of all original media packets of the transmission FEC block is the reception of exactly the same number (*M*) of packets (of any type: media or redundant) as there were original media packets in the block.

In order to compute the proper transmission block size FEC_p, we must fix a desired Decoding Error Rate (DER), i.e. the acceptable decoding failure probability at the receiver.

According to the binomial distribution, equation (2) gives the decoding failure probability δ at the packet loss rate *p* if the FEC block size is equal to *N*.

$$\delta = \sum_{n=N-M+1}^{N} \binom{N}{n} \cdot p^n \cdot (1-p)^{N-n} \qquad (2)$$

For computing the carrier block's minimal length for a satisfactory communication, it is sufficient therefore to steadily increase in equation (2) the carrier block length *N* until the desired decoding error rate (DER) is met.

In real-time streaming there is a tradeoff between the number of media packets *M* and the cost of FEC overhead. Before playing the media, the receiver must hold in the buffer enough packets to restore the recoverable losses. The receiving side of the media application is already equipped with a playback buffer in order to compensate for the network jitter and to reorder packets arriving in the wrong order. The playback buffer must be large enough to also hold packets of the FEC block (at least *M* packets for an MDS code). For example in VOIP with a 20 ms sampling rate (g729r8 or AMR codec) the number of media packets in a single FEC block must not exceed 20 – 25 packets (each carrying one sample).

III. Capillary routing

Capillary routing seeks to minimize the impact of individual link failures on real-time streaming, requiring thus from the encoder a fewer effort for recovering the failure.

The strategy for capillary routing can be best defined by describing an iterative Linear Programming (LP) process transforming a simple single-path flow into a capillary route. First minimize the maximal value of the load of links by minimizing an upper bound value applied to all links. By balancing so the maximal load values for all links, in the first layer, the full mass of the flow is split equally across the available parallel routes. Then, find the bottleneck links of the first layer. By maintaining the first upper bound (applied to all links) on its minimal level, minimize the maximal load of the remaining links by minimizing a new upper bound value applied to all links except the bottleneck links of the first layer. This second iteration discovers the sub-routes and the sub-bottlenecks of the second layer. Then, minimize the maximal load of the remaining links, now also without the bottlenecks of the second layer (maintaining the first and second upper bounds at their lowest level), and continue the iteration until the entire footprint of the flow is discovered. A flow traversing a large dense network with hundreds of nodes may have hundreds of capillary routing layers.

Although this iterative LP process defining the capillary routing is completely valid, the precision errors propagating through the LP minimization statements of the layers of capillary routing reach however noticeable sizes and, when dealing with tiny loads, often result in numerical instabilities (and infeasible problems). We have found a different, stable LP method maintaining the values of parameters and variables always in the same scale.

Instead of decreasing the maximal value of loads of the links, the routing path is discovered by solving the max flow problem. The resulting routing solutions of these two methods are identical except that the proportions of flow differ by the increase factor of the max flow solution. The diagrams of Fig. 1 to Fig. 3 present on a network example with 6 nodes the discovery of the first three layers of capillary routing according to the max-flow LP approach.

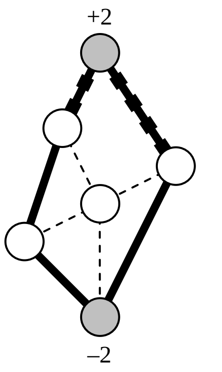

Fig. 1. Maximize the flow of the initial problem with one source and one sink, fix the new flow-out coefficients at the nodes and find the bottleneck links (layer 1)

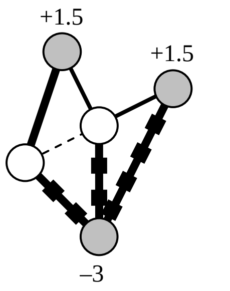

Fig. 2. Remove the bottleneck links from the network, adjusting the flow-out coefficients at the adjacent nodes, maximize the flow in the new sub-problem, fix the new flow-out coefficients at the nodes and find the new bottlenecks (layer 2)

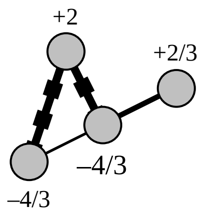

Fig. 3. Again remove the bottleneck links of the previous solution from the network, again maximize the flow in the obtained new problem, fixing the new flow-out coefficients and find the new bottlenecks (layer 3)

The max-flow problem is defined by the flow-out coefficients at each node. Initially only the peer nodes have non-zero flow-out coefficients: +1 for the source and –1 for the sink. At each subsequent layer we have a bounded multiple-sources/multiple-sinks problem: a uniform flow from a set of sources to a set of sinks, where all rates of transmissions by sources and all rates of receptions by sinks increase proportionally in respect to each node's flow-out coefficient (either positive or negative). In the above example (Fig. 1 to Fig. 3) the flow increase factor of the network at the first layer (F^1) is equal to 2 (Fig. 1), the network flow increase factor at the second layer (F^2) is equal to 1.5 (Fig. 2) and the flow increase factor of the third layer (F^3) is 4/3 (Fig. 3).

The LP problem at each successive layer is obtained by complete removal of the bottlenecks from the previous LP problem, adjusting correspondingly the flow-out coefficients of the adjacent nodes (to respect the flow conservation rule) and thus possibly producing new sources and sinks in the network. Except for the single-source/single-sink problem of the first layer, the successive layer problems (bounded multiple-sources/multiple-sinks problems) do not belong in general to the simple class of "network linear programs" [12].

Let us conclude with equations the construction of successive problems. We define the bounded multiple-sources/multiple-sinks problem at layer *l* by sets of nodes and links and by parameters for sources and sinks (all indexed with an upper index *l*) as follows:

- set of nodes N^l,
- set of links $(i,j)\in L^l$, where $i\in N^l$ and $j\in N^l$,
- and flow-out values f_i^l for all $i\in N^l$

At layer *l* the max-flow solution yields the flow increase factor F^l and the set of bottlenecks B^l, where $B^l\subset L^l$.

The sets N^{l+1}, L^{l+1} and the parameters f^{l+1} of the next layer are computed according the following equations:

- $N^{l+1}=N^l$
- $L^{l+1}=L^l-B^l$
- $f_j^{l+1}=f_j^l\cdot F^l \quad + \underbrace{\sum_{(i,j)\in B^l}(+1)}_{\text{add 1 for each incoming bottleneck link } (i,\, j)} \quad + \underbrace{\sum_{(j,k)\in B^l}(-1)}_{\text{subtract 1 for each outgoing bottleneck } (j,\, k)}$

After a certain number of applications of the max-flow objective with corresponding modifications of the problem, we finally obtain a network having no source and sink nodes. At this moment the iteration stops. All links followed by the flow in the capillary routing are enclosed in bottlenecks of one of the layers.

To restore the original proportions of the flow, the flow increases by the preceding max-flow solutions must all be compensated. The true value of flow $r_{i,j}$ traversing the bottleneck link $(i,j)\in B^l$ of layer *l* is the initial single unit of flow divided by the product of the flow increase factor F^l of layer *l* with the flow increase factors F^i (where $1\le i<l$) of all preceding layers:

$r_{i,j}=\dfrac{1}{\prod_{i=1}^{l}F^i}$, where *l* is the layer for which $(i,j)\in B^l$

The max-flow approach proves to be very stable, because it maintains all values of variables and parameters within a close range of unity (even for very deep layers with tiny loads) and also because it enables to validate and if necessary re-calibrate the flow-out parameters of the LP problem formulated for the next layer of capillary routing. Re-calibration of parameters before solving the LP problem at each layer avoids undesirable propagation of errors leading to numerical instabilities.

IV. Bottleneck hunting loop

Bottlenecks of each max-flow solution are discovered in a bottleneck hunting loop. Each iteration of the hunting loop is an LP cost minimizing problem that reduces the load of the traffic over all links having maximal load and being suspected as bottlenecks. Only links maintaining their load at the initial maximal level will be passed to the next iteration. Links whose load has been reduced under the LP objective are not bottlenecks and removed from the list of candidates. The bottleneck hunting iteration stops if there are no more links to remove.

Let us demonstrate bottleneck hunting on an example of three transmitting nodes and two receiving nodes in Fig. 4. The flow of the optimal solution can be proportionally increased at most by a factor of 4/3, such that each flow-out coefficient at sources become equal to 4/3 and each flow-in coefficient at sinks become equal to -2. Before starting bottleneck hunting, we constrain the LP model by fixing the flow increase factor on its maximal value of 4/3. A possible max-flow solution is shown in Fig. 5. The bottleneck links are among the four maximally loaded candidate links {*a*, *b*, *d*, *e*} marked by thick dashes. To reduce the search space we minimize the sum of loads of four candidate links {*a*, *b*, *d*, *e*}.

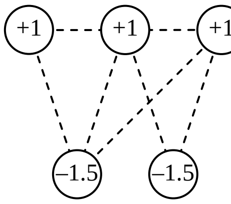

Fig. 4. An example of a bounded multiple-sources/multiple-sinks problem (obtained during construction of the capillary routing from a network with one source and one destination nodes)

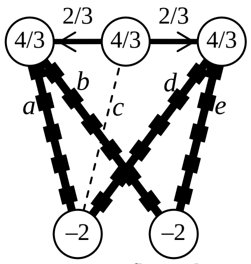

Fig. 5. A max-flow solution with the flow increase factor of 4/3, containing four maximally loaded candidate links {*a*, *b*, *d*, *e*}

The initial value of this total cost is equal to 4 in Fig. 5. The minimal value it can be reduced to is 3, and a possible solution is shown in Fig. 6. It is obtained by delegating the flow from one of four candidate links to a non-bottleneck portion of the network. The link *d*, the one which did not maintain its load on the maximum, is not a true bottleneck link and is therefore removed from the list. The total cost of the remaining three candidates {*a*, *b*, *e*} is equal to 3.

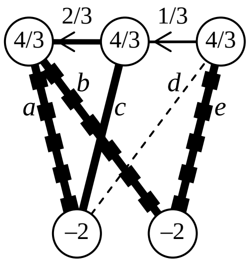

Fig. 6. Cost reduction applied to four fully loaded links of Fig. 5 reduced the load of a candidate link *d*, and the candidate links are now {*a*, *b*, *e*}.

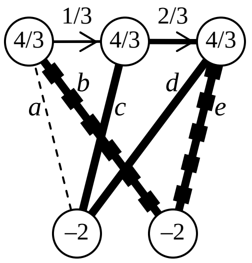

Fig. 7. Cost reduction applied to the three fully loaded links of Fig. 6 reduced the load of another candidate link *a*, and the true bottleneck links are {*b*, *e*}.

By minimizing a second time the cost function, now applied on the three candidates, we delegate out the load from yet another link reducing the total cost to the minimum equal to 2 (see Fig. 7). The two remaining links {*b*, *e*} maintaining their loads always at the maximum, are the true bottleneck links. Further applications of the LP cost objective can not result in additional reduction of the total load of these two bottleneck links.

During this bottleneck hunting example non-candidate links also had maximal loads (link *d* in Fig. 5 and Fig. 7, link *c* in Fig. 6 and Fig. 7), but they are not bottlenecks, since there were solutions, for each of them, where their load was not at the maximum.

For the previous example the two bottlenecks were found in two iterations, for larger networks with hundreds of nodes, hunting of bottlenecks of max-flow solutions may require dozens of iterations at each layer of capillary routing.

Fig. 8 shows an example of capillary routing, built simultaneously on 200 independent unbounded networks with 300 nodes in each (total 60,000 nodes and 511,140 links). We show the decrease in the average number (over all unbounded networks) of suspected links as bottleneck hunting of a given capillary routing layer progresses. Bottleneck hunting of the presented example requires from 14 to 23 iterations, depending on the layer of the capillary routing (from 1 to 10).

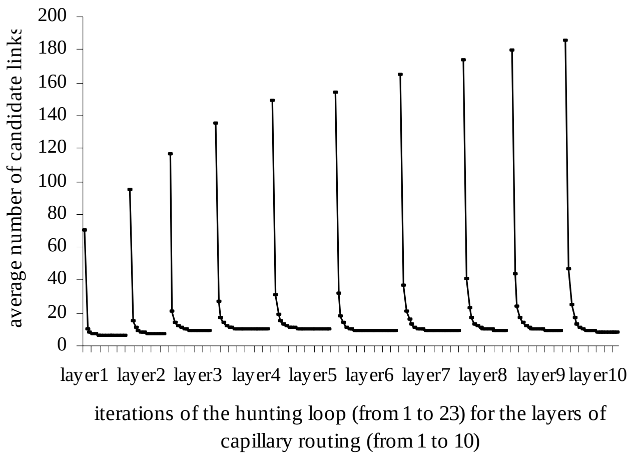

Fig. 8. Evolution of the number of candidate links during bottleneck hunting loops, for 10 capillary routing layers on a problem with 200 unbounded networks

Each iteration of the bottleneck hunting removes from the list of candidates numerous non-bottleneck links. At the end of each hunting loop the list of candidates contains only true bottleneck links. The average number of bottleneck links at each layer is between 5.9 and 9.9.

V. Friendliness of Capillary Routing

We compute the average FRROR rating for various network samples in order to evaluate the overall performance of the capillary approach. First, we consider the first layer routing scheme for each considered network sample and obtain thus the average FRROR rating for all routing (max-flow) schemes of the first layer. Then we compute the second layer routing individually for each considered network sample and obtain the average FRROR rating for the routing suggestions of the second layer. We measure the average FRROR rating for the capillary routing layers 1 to 10 and show the increase in the layer number grows.

In Fig. 9, we have eight sets of network samples, each containing 25 network samples. At the same time we consider also 8 media streams which differ by their static tolerance to losses varying from 3.3% to 7.5%. Thus for each set we have 8 curves of average FRROR ratings. All of them decrease as the capillary routing layer increases from layer 1 to 10 demonstrating the improvements due to the stronger capillarization.

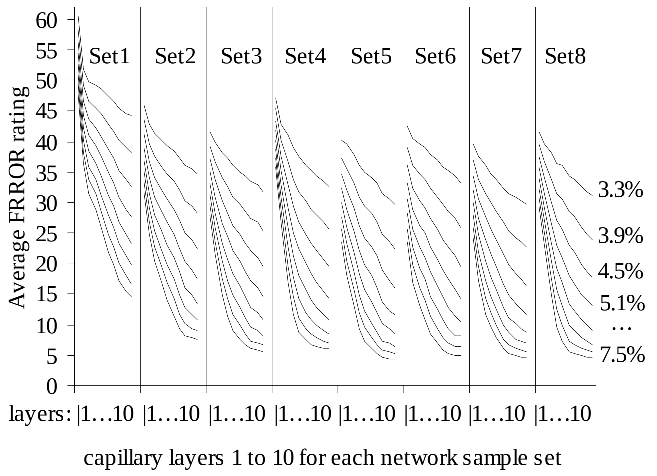

Fig. 9. Average FRROR as a function from the capillary routing layer (the static tolerance of the stream from 3.3%, for the upper curves, to 7.8%, for the lower curves, by a step of 0.6%)

Although spreading out of the flow uses more links and therefore also increases the total rate of failures in the communication footprint, capillarization of the basic multi-path routing also through the non-bottleneck portions of the network reduces however the total FRROR rating considerably and therefore also the FEC effort of the sender combating the link failures and packet losses.

Logically, the FRROR curve of the media stream is shifted down as the statically added tolerance increases. At the same time it is interesting to observe that, in contrast to a weak static tolerance, the presence of a higher static tolerance yields a much stronger efficiency gain achieved by the deeper routing layers.

The pattern of the FRROR curve, as a function of the layer, depends on the distance between the peers, the network size and its density. The network samples for the above chart are drawn from a random walk wireless Mobile Ad-hoc Network (MANET). Initially the nodes are randomly distributed on a rectangular area, and then at every timeframe they move according to a random walk algorithm. If two nodes are close enough (and are within the coverage range) then there is a link between them. In the above example, there are 300 nodes and 200 time-frames, each leading to a separate network sample (all of which are distributed into eight sets represented on the above chart).

The FRROR rating of routing samples is computed by equation (1), where the FEC block size (as function of packet loss rate p) is computed based on equation (2). The number of media packets (M) per transmission block is 20 and the desired decoding failure rate (DER) is 10^{-5}.

VI. Conclusions

We introduce multi-path capillary routing, built layer by layer. The first layer provides a simple max-flow solution, but as the layer number increases the spreading of the underlying routing scheme makes the network more secure for real-time media streaming. We introduce FRROR, a method for rating multi-path routing schemes by a single scalar value. The FRROR rating corresponds to the total redundancy overhead that the sending node provides in order to combat the losses occurring from non-simultaneous failures of links in the communication path. By rating the FEC friendliness we show a substantial improvement of the routing topology and increase of its FEC efficiency, as the routing capillarizes.

Capillary fault-tolerant routing can be applicable to Ad-Hoc or sensor networks, to mobile networks, where wireless content can be streamed to and from the user via multiple base stations; or to the public internet, where, if the physical routing cannot be accessed, path diversity can be still obtained relying on overlay networks using peer-to-peer relay nodes [11] and [13].

VII. References

[1] David J. C. MacKay, “Fountain codes”, IEE Communications, Vol. 152 Issue 6, Dec 2005, pp. 1062-1068

[2] Amin Shokrollahi, “Raptor codes”, ISIT’04, Jun 27 - Jul 2, p. 36

[3] Michael Luby, “LT codes”, FOCS’02, November 16-19, pp. 271-280

[4] Ingemar Johansson, Tomas Frankkila, Per Synnergren, “Bandwidth efficient AMR operation for VoIP”, Speech Coding 2002, Oct 6-9, pp. 150-152

[5] Yicheng Huang, Jari Korhonen, Ye Wang, “Optimization of Source and Channel Coding for Voice Over IP”, ICME’05, Jul 06, pp. 173-176

[6] Chinmay Padhye, Kenneth J. Christensen, Wilfrido Moreno, “A new adaptive FEC loss control algorithm for voice over IP applications”, IPCCC’00, Feb 20-22, pp. 307-313

[7] Eitan Altman, Chadi Barakat, Victor M. Ramos, “Queueing analysis of simple FEC schemes for IP telephony”, INFOCOM 2001, Vol. 2, Ap 22-26, pp. 796-804

[8] Qi Qu, Ivan V. Bajic, Xusheng Tian, James W. Modestino, “On the effects of path correlation in multi-path video communications using FEC over lossy packet networks”, GLOBECOM’04, Vol. 2, Nov 29 - Dec 3, pp. 977-981

[9] Tawan Thongpook, “Load balancing of adaptive zone routing in ad hoc networks”, TENCON 2004, Vol. B, Nov 21-24, pp. 672-675

[10] Rui Ma, Jacek Ilow, “Regenerating nodes for real-time transmissions in multi-hop wireless networks”, LCN’04, Nov 16-18, pp. 378-384

[11] Thinh Nguyen, P. Mehra, Avideh Zakhor, “Path diversity and bandwidth allocation for multimedia streaming”, ICME’03 Vol. 1, Jul 6-9, pp. 663-672

[12] Robert Fourer, *A modeling language for mathematical programming*, Thomson – Brooks/Cole, second edition, 2003, p. 343

[13] Tuna Guven, Chris Kommareddy, Richard J. La, Mark A. Shayman, Bobby Bhattacharjee “Measurement based optimal multi-path routing”, INFOCOM 2004, Vol. 1, Mar 7-11, pp. 187 - 196