Fault-tolerant multi-path routing for real-time streaming with erasure resilient codes

Emin Gabrielyan
EPFL and Switzernet Sàrl
Lausanne, Switzerland
emin.gabrielyan@switzernet.com

Roger D. Hersch
École Polytechnique Fédérale de Lausanne (EPFL)
Switzerland
rd.hersch@epfl.ch

Abstract

Thanks to large buffers, Forward Error Correction (FEC) improves the reliability of off-line streaming significantly. Real-time streaming however puts hard restrictions on the buffer size leaving FEC helpless for combating long link failures on a single path route. Compared with buffering, multi-path routing is another orthogonal method, which can make FEC efficient also for real-time streaming. We introduce a capillary routing algorithm offering multi-path routing topologies of increasing path diversity. We also propose a scalar value called Redundancy Overall Requirement (ROR) for measuring the friendliness of a multi-path routing pattern toward FEC. A dozen of capillary routing suggestions, built on several hundreds of network samples obtained from a simulation of a wireless random walk Mobile Ad-Hoc Network (MANET), are rated with ROR. We show that the sender's channel coding effort decreases substantially as the spreading of the routing grows.

Keywords: *Capillary routing, multi-path routing, fault-tolerance, real-time streaming, Voice over IP, erasure resilient codes*

1. 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, applied at the packet level, can mitigate packet losses.

Thanks to the unlimited buffering time of off-line applications Forward Error Correction (FEC) yields spectacular results. With Raptor codes [1], via a satellite broadcast channel without feedbacks, it is possible to recurrently deliver voluminous GPS maps to millions of motor vehicles under conditions of arbitrary fragmental visibility. In the film industry, LT codes [2] 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 (otherwise impossible with TCP file transfer, converging the bandwidth, irrespectively to the effective capacity, to a function from the packet loss rate and RTT, unavoidably high due to distance). Third Generation Partnership Project (3GPP), recently adopted Raptor as a mandatory code in Multimedia Broadcast/Multicast Service (MBMS), for its significant performance in file transfer and its low CPU cost [3], [4], [5].

The above examples of off-line streaming can 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 of a very short life time 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 from applications of FEC to real-time streaming. In [6] it has been shown improvements from the application of FEC only if the stochastic packet losses range is between 1% and 5%. For real-time packetized streaming the author of [7] proposed to combine FEC with retransmissions. In [8] a high overhead has been reported from the use of FEC during bursts. The author of [9] claims that for two-way, delay-sensitive real-time communications, the application of FEC on the packet level can not give any valuable results at all.

Studies stressing the poor FEC efficiency always assumed that the real-time streaming follows a single path. There is an emerging body of a literature showing that the path diversity makes FEC applicable also in real-time streaming [10], [11], [12], [13], [14] and [15]. In all these studies the path diversity is limited to either two (possibly correlated) paths or in the best case to a sequence of parallel and serial links. The routing topologies, so far, were not regarded as a space of possibilities and a ground for searching a FEC effective pattern. In this paper we try to present a comparative study for a wide range of multi-path routing patterns. Single path routing, being considered as too hostile, is excluded from our comparisons.

As a method for generating multi-path routing patterns of various path diversity, we propose *capillary routing* algorithm. Capillary routing is built layer by layer, providing at each layer a multi-path routing suggestion spreading out across more links as the layer number increases. The first layer is a simple multi-path

routing representing a max-flow solution. Successive layers are obtained by recursively spreading the individual sub-flows of previous layers. With the capillarization of the routing, the communication uses the network more reliably employing more of alternative transmission capacities offered by the network topology. The last layer represents the complete capillary routing and. Contrary to the shortest path or max-flow solutions, the complete capillary routing has one unique solution for any given network and pair of peers. We present the capillary routing construction in sections 2 and 3.

To compare multi-path routing suggestions, we relay on a virtual sender dynamically adapting the amount of transmitted FEC codes in responce to the changes in packet loss rate measured at the receiver. Adaptive FEC in real-time streaming was already proposed by several other authors [16], [17], [8], [6] and [7]. The end-to-end adaptive FEC mechanism is not aware of the underlying routing and is implemented entirely at the application level of the end nodes. By default the sender is streaming the media always with a constant FEC, tolerating a certain minimal packet loss rate. Packet loss rate is measured at the receiver and is constantly reported back to the sender with the opposite flow, usually using Real-time Transport Control Protocol (RTCP). The sender always increases the FEC overhead whenever the packet loss rate is about to exceed the tolerable limit. The friendliness of a multi-path routing pattern toward FEC is rated by Redundancy Overall Requirement (ROR), which is the total amount of the adaptive redundancy required from the sender during the communication time (section 4). The novelty brought by ROR, is that a routing topology of any complexity can be rated by a single scalar value.

In section 5, we evaluate the advantageousness of the capillarization by rating with ROR each layer of capillary routing. Network samples for the measurements are obtained from a wireless random walk Mobile Ad-hoc Network (MANET) with several hundreds of nodes. We show that by evolving the path diversity provided by the first layer of the capillary routing (i.e. the max-flow solution) towards more elaborated multi-path routing we still can reduce significantly the overall channel coding effort of the sender. Multi-path routing ax alone, similarly to the buffering ax, can substantially burst the efficiency of FEC.

2. Capillary routing

Capillary routing aims at minimizing the impact of individual link failures to real-time stream giving thus the encoder a greater chance to recover the failure.

The capillary routing algorithm is defined by an iterative LP process transforming a simple single-path flow into a capillary routing. First minimize the maximal value of the load of all links by minimizing an upper bound value applied to all links. Find the bottleneck links of the first layer. Maintaining the first upper bound (applied to all links) on its minimal level, minimize the maximal load of remaining links by minimizing another upper bound value applied to all links except the bottleneck links of the first layer. At the second iteration we discover the sub-routes and the sub-bottlenecks of the second layer. Minimize the maximal load of the remaining links, now without also 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.

Fig. 1 to Fig. 3 show three layers of the capillary routing on a toy network example. The top node on the diagrams is the sender and the bottom node is the receiver; all links are oriented from top to bottom.

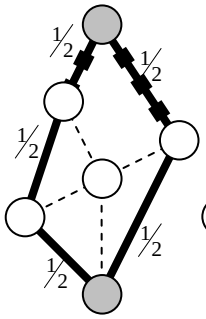

Fig. 1. In the first layer the flow is equally split over two paths, two links of which, marked by thick dashes, are the bottlenecks.

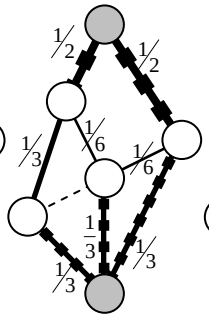

Fig. 2. The second layer minimizes the maximal load of the remaining seven links and finds three bottlenecks, each with a load of 1/3.

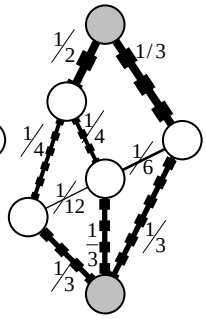

Fig. 3. The third layer minimizes the maximal load of the remaining four links and finds two bottlenecks each with a load of 1/4.

Although the iterative LP method used for the definition of the capillary routing is fully valid; precision errors propagating through the sequence of LP minimizing problems often reach noticeable sizes and, when reaching tiny loads, result in numerical instabilities and infeasible LP problems. We have found a different, stable LP method maintaining the parameters and variables always in the same order of grandeur.

Instead of decreasing the maximal value of loads of links, the routing pattern is discovered by solving the max flow problem. The resulting paths of these two methods are identical except that the proportions of flow are different by the increase factor of the max flow solution.

The max-flow problem is defined by the flow-out coefficients at each node. At each layer we have 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 must change

proportionally according to the each node's flow-out coefficient (either positive or negative). The objective is to maximize the flow respecting the proportionality bounds of all sources and sinks.

At the first layer only the peer nodes have non-zero flow-out coefficients: +1 for the source and –1 for the sink. 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 the unicast problem of the first layer, the successive layers do not belong to the simple class of "network linear programs" [18].

For the example of Fig. 1 to Fig. 3 the diagrams of Fig. 4 to Fig. 9 now show the capillary routing discovery with the stable max-flow approach. Fig. 4 and Fig. 5 show the max-flow solution and the bottlenecks of the first layer.

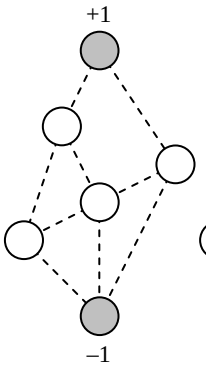

Fig. 4. Initial problem with one source and one sink node

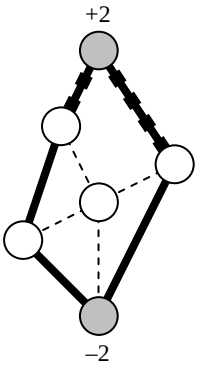

Fig. 5. Maximize the flow by a factor of 2, fix the new flow-out coefficients at the nodes and find the bottleneck links

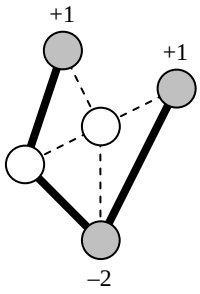

Fig. 6. Remove the bottleneck links from the network adjusting the flow-out coefficients at the adjacent nodes

Fig. 6 to Fig. 7 show the second layer and Fig. 8 to Fig. 9 show correspondingly the discovery of the third layer.

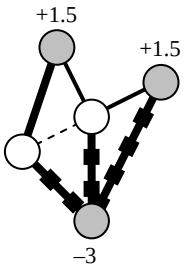

Fig. 7. Maximize the flow in the new sub-problem (by a factor of 1.5), fix the new flow-out coefficients at the nodes and find the new bottlenecks

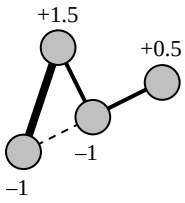

Fig. 8. Remove the bottleneck links from the network and adjust correspondingly the flow-out coefficients at the adjacent nodes

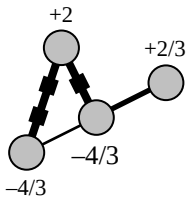

Fig. 9. Again maximize the flow in the obtained new problem (by a factor of 4/3) and fix the new resulting flow-out coefficients at the nodes

Let us present the iterations in equations. We define the max-flow problem at layer l by sets of nodes and links and by flow-out parameters for sources and sinks (indexed all 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 flow-out parameters f^{l+1} of the next layer are computed according the following equations:

- $N^{l+1} = N^l$
- $L^{l+1} = L^l - B^l$
- $f_j^{l+1} = f_j^l \cdot F^l + \sum_{(i,j)\in B^l} (+1) + \sum_{(j,k)\in B^l} (-1)$

add 1 for each incoming bottleneck link (i, j) — 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 be all compensated. The true value of flow $r_{i,j}$ traversing the bottleneck link $(i, j) \in B^l$ of layer l is the initial one unit of flow divided by the product of the flow increase factors of layer l with all preceding layers:

$r_{i,j} = \frac{1}{\prod_{i=1}^{l} F^i}$, 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 re-calibrate the LP problem before parsing to the next layer avoiding thus the undesirable propagation of precision errors.

In the next section we present the bottleneck identification algorithm.

3. 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 traffic over all links being suspected as bottlenecks. Only links maintaining their load at the initial level will be passed to the next

iteration of the hunting loop. Links undergoing to load reduction under the LP objective are removed from the suspect list and the bottleneck hunting iteration stops if there are no more links to remove.

Let us show the bottleneck hunting on the example of Fig. 10 with three transmitting nodes and two receiving nodes. The flow 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 (see Fig. 11). The bottleneck links are among four maximally loaded suspected links, marked in Fig. 11 by thick dashes.

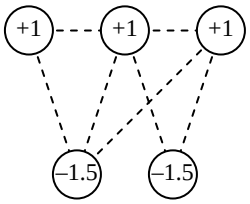

Fig. 10. A sample multiple-source multiple-sink max-flow problem (obtained during construction of the capillary routing from a network with one source and one destination nodes)

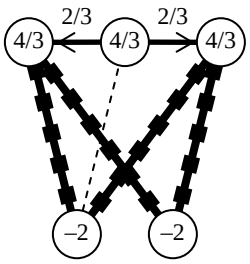

Fig. 11. The flow increase factor of the max-flow solution is 4/3; there are four maximally loaded links

An LP cost-minimizing objective can reduce the sum of loads of all four suspected links from the initial value of 4 to the minimal value of 3 (see Fig. 12). In this min-cost solution one of four suspected links does not maintain its load on the maximum and thus there left only three suspected links, marked in Fig. 12 by thick dashes.

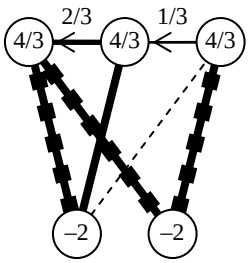

Fig. 12. Cost reduction applied to four fully loaded links of Fig. 11 delegates completely the load of one suspected link to the network portion outside of the suspect list

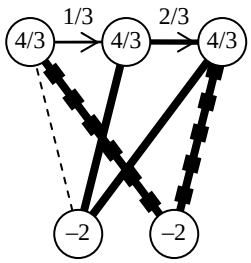

Fig. 13. Cost reduction applied to three fully loaded links of Fig. 12 delegates completely the load of one suspected link to non-suspected portion of the network

The sum of loads of now three suspected links can be further reduced from the value of 3 to a minimal value of 2 (see Fig. 13). Now there are only two links maintaining their loads at the maximal value, marked in Fig. 13 by thick dashes. These two remaining links are the true bottleneck links, since additional applications of the LP cost minimizing objective does not result in any further reduction of the sum of loads.

In this example the two bottlenecks were found in two iterations, for larger networks with hundreds of nodes the bottleneck hunting of each capillary routing layer can take dozens of iterations.

For capillary routing patterns, built simultaneously on 200 unbounded networks samples with 300 nodes in each (total 60,000 nodes and 538,940 links), Fig. 14 shows the decrease of the number of suspected links during the bottleneck hunting loop of each capillary routing layer. Depending on the layer of the capillary routing (1 to 10), the bottleneck hunting loop takes 10 to 25 iterations.

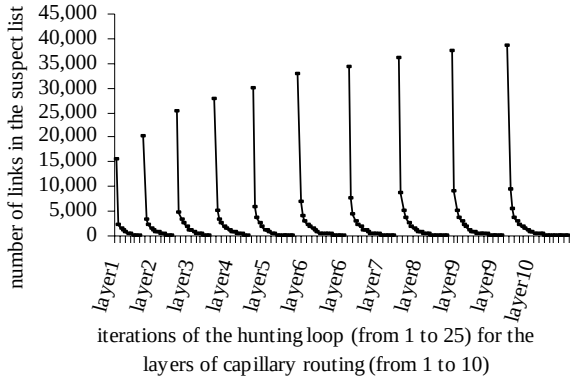

Fig. 14. Decrease of the number of suspected links during bottleneck hunting loops of 10 capillary routing layers (total number of links in 200 independent networks)

Each iteration of the bottleneck hunting removes from the suspect list hundreds of non-bottleneck links. As soon as for a particular unbounded network bottlenecks are identified they are also removed from the suspect list. Thus all curves hit zero at the end of each hunting loop. The total number of found bottlenecks for all 200 unbounded networks at each layer is between 1,243 and 1,953.

4. Redundancy Overall Requirement (ROR)

Spread routing alone would not solve the problem of tolerance without FEC. If media stream is not capable to sustain any losses at all, by spreading the transmission the media becomes even more vulnerable, since there are more links whose failures will damage the stream. Most real-time media streaming applications are tolerant to a certain level of packet losses due to passive error concealment or media encoding techniques (e.g. a packet may carry duplicates of media from previous slots, encoded with a lower rate source coding, etc). 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 a little static tolerance, combating weak failures, with a dynamically added adaptive FEC, combating the strong failures exceeding the tolerable packet loss rate.

ROR is defined as the sum of all transmission rate increase overheads needed for combating all individual link failures of the multi-path routing pattern. For

example if the communication footprint consists of five links and in response to each link failure the sender increases the packet transmission rate by 25%, then ROR will be equal to the sum of these five rate increase overheads, i.e. to $ROR = 5 \cdot 25\% = 1.25$.

Let *P* be the usual packet transmission rate and P_l be the increased rate of the sender, responding to the failure of a link $l \in L$, where *L* is the set of all links. Then:

$$ROR = \sum_{l \in L} \left(P_l \big/ P - 1 \right) \quad (1)$$

Let us consider a long communication, and let *D* be the total failure time of a single network link during the whole duration of the communication (which is the product of the duration of a single link failure, the frequency of a single link failure and the total time of the communication). Then according to equation (1) $D \cdot P \cdot ROR$ is the number of extra redundant packets that the sender will transmit in order to compensate all individual link failures occurring during the whole communication time (assuming a single link failure at a time and a uniform probability and duration of link failures). For various multi-path routing suggestions parameters *D* and *P* remain constants. Therefore, ROR is the proportionality coefficient of a given multi-path routing pattern for the overall number of extra redundant packets transmitted by the sender during the communication. The smaller the ROR rating of a multi-path routing pattern, the fewer are the redundant packets required from the sender.

Redundant packets are injected in the original stream for every block of *M* source packets using systematic erasure resilient codes (thus without affecting the original media packets). During the streaming, the number of source packets (*M*) is supposed to stay constant. The number of redundant packets for each block of *M* source packets is however variable, depending on the conditions of the erasure channel. The *M* media packets with all related redundant packets form a FEC block. By FEC_p 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 a static tolerance to losses $0 \le t < 1$ by maintaining constant FEC code. Therefore by default the packets are streamed in 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 failure (or congestion) time of a link in the communication path, is the portion of the traffic being still routed toward the faulty route. Thus a complete failure of a link *l* carrying according to the routing pattern a relative traffic load of $0 \le r(l) \le 1$ will produce at the receiver a random packet loss rate equal to the same relative traffic load $r(l)$. The equation (1) can thus be re-written as follows:

$$ROR = \sum_{l \in L \mid t \le r(l) < 1} \frac{FEC_{r(l)} - FEC_t}{FEC_t} \quad (2)$$

a sum over all links carrying a flow exceeding the tolerable loss limit, except the links passing the entire traffic

According the sum's index in above equation (2), the critical links of the path carrying the entire traffic are ignored. The FEC required for the compensation of long failures of such links would be infinite. All considered multi-path routing schemes by construction (sections 2 and 3) are delegating the load from a critical route over other links, if of course the network topology makes it possible. If the link is critical by the network topology any routing scheme unavoidably will pass its entire traffic through such a link, then without a risk of affecting the comparison results such a link can be removed from all suggested routings in order to compare their remaining portions.

We compute the FEC_p function assuming a Maximum Distance Separable (MDS) code, e.g. Reed-Solomon code. By the choice of an MDS code, the reception of as many transmitted packets as there were source media packets in the block (*M*) is the only condition for the successful decoding of all original media packets.

In order to compute the transmission block size $FEC_p \ge M$ as a function of the packet loss rate *p*, we must fix the desired Decoding Error Rate (DER), i.e. the acceptable decoding failure probability at the receiver.

To collect mean *M* packets at the receiver we must transmit $M \big/ 1-p$ packets at the sender. However the probability of receiving $M-1$ packets or $M-2$ packets (which makes the decoding impossible) is quite high. Therefore for transmitting blocks carrying small number (*M*) of media packets, we must send much more redundant packets in the block than is necessary to receive mean *M* packets at the receiver side. The average number of received packets should be maintained much higher than *M* and the probability δ of receiving less than *M* packets must be maintained very low: $\delta \le DER$.

The probability of having $N-M+1$ or more losses (i.e. less than *M* survived packets), is computed according to the binomial distribution as follows:

$$\delta = \sum_{n=N-M+1}^{N} \binom{N}{n} \cdot p^n \cdot q^{N-n}, \quad \text{where } \binom{N}{n} = \frac{N!}{n! \cdot (N-n)!} \text{ and } q = 1-p \quad (3)$$

The above formula gives the decoding failure probability with an MDS erasure resilient code if the FEC block size is equal to *N*. Therefore for computing the carrier block's minimally needed length for a satisfactory communication, it is sufficient to steadily increase the carrier block length *N* until the desired decoding error rate (DER) is met.

Several FEC_p / M functions (FEC overheads) for media packets from 1 to 10 are plotted in Fig. 15 (for $DER = 10^{-5}$). These functions are bounded below by the theoretical limit $1/(1-p)$. Higher the number of media packets in the block (i.e. longer the buffering time), closer the rate increase factor can approach to the theoretical limit.

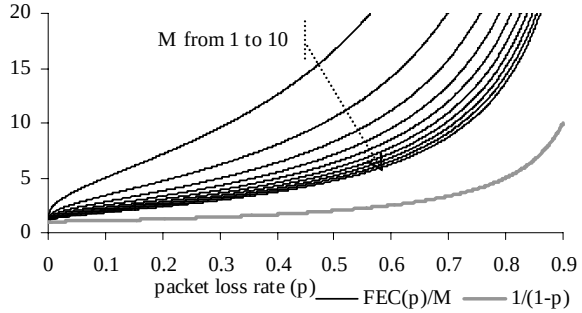

Fig. 15. FEC overhead as a function from the packet loss rate ($DER = 10^{-5}$)

In real-time streaming there is a tradeoff between *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 to compensate the network jitter and to reorder packets arriving in wrong order). Keeping too many packets in the playback buffer may exceed the tolerable latencies. For example in VOIP with 20 ms sampling rate (g729r8 or AMR codec) the number of source media packets in a single FEC block must not exceed 20 – 25 packets. The next section presents ROR rating of numerous multi-path routing patterns for real-time streaming.

Path diversity can be required also in off-line streaming for example for downloading at the maximal rate of the last kilometer bottleneck. In order to combat the arbitrary losses arising in different network locations the sender, with multi-path routing, can temporarily increase its transmission rate and maintain the constant feeding of the last kilometer bottleneck (see streaming of one-way video from multiple servers [14] and [20]). In off-line applications, the number *M* of source packets in a FEC block can be very large. The buffering time can be a few minutes long corresponding to thousands of source media packets. With capacity approaching fountain codes [19] $FEC_p = M/(1-p)$. Therefore for off-line streaming the equation (2) can be re-written as follows:

$$ROR = \sum_{l \in L \mid t \le r(l) < 1} \left(\frac{1-t}{1-r(l)} - 1 \right) \quad (4)$$

5. Measuring the friendliness of capillary routing

In order to evaluate the overall performance of the capillary routing approach we rate with ROR capillary routing patterns on various network samples. First, we rate the first capillary routing layer on each of selected network samples obtaining the average ROR rating of the first capillary routing layer. Similarly we rate higher layers of the capillary routing using the same set of network samples.

In Fig. 16, we have eight sets, each containing 25 network samples. We compute the average ROR rating as a function of the capillary routing layer (1 to 10) for each of eight sets. We consider also 16 media streams differing by their static tolerance to losses from 3.3% to 7.8%. Thus for each set we have 16 curves of average ROR ratings. All of them decrease as the capillary routing layer increases demonstrating the improvements provided by stronger capillarization.

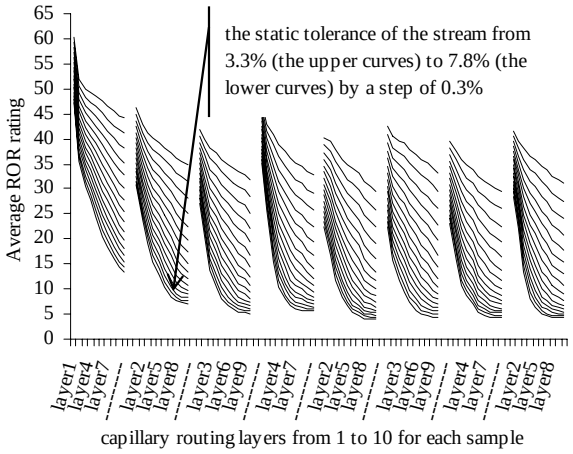

Fig. 16. Average ROR as a function from the capillary routing layer

Logically, the ROR curve of the media stream is shifted down by every unit of the statically added tolerance. At the same time it is interesting to observe that the strong static tolerance stresses the efficiency gain achieved by the deep layers much better than the weak static tolerance does. Although there are hundreds layers in the complete capillary routing, the first few layers alone reduce the average FEC effort of the sender by a factor of four.

Of course, the exact pattern of the ROR 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 obtained from

simulations of 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 timeframes each leading to a different network sample (all of which are broken into eight sets represented on the above chart). The number of media packets per each transmission block (M) is 20 and the desired decoding failure rate (DER) is 10^{-5}. ROR is computed according to equation (2) and FEC_p function according to equation (3).

Multi-path routing suggestions for fault-tolerant streaming are applicable not only to Ad-Hoc or sensor networks, but also 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 improved, a near capillary routing can be still obtained through an overlay network using peer-to-peer relay nodes or media routers (see [15] and [21]). Spreading of the flow from the User Agents (UA) to the media routers can be implemented in the firmware of a SIP phone, in the NAT router of the user or in the closest SIP proxy.

Spreading can be obtained more transparently at a lower network layer using VPN tunnels such that the flow at the source is split across VPN interfaces each leading to various VPN gateways scattered across the network. Alternatively, path diversity can be obtained also by assigning to media gateways or SIP servers more than one IP interfaces, each connected to a different ISP.

6. Conclusion

We introduce multi-path capillary routing, which is built layer by layer. The first layer provides a simple max-flow solution, but as the layer number increases the underlying routing spreads out, relying on the network more securely. We also introduced ROR, a method for rating a multi-path routing by a single scalar value. The ROR rating corresponds to the total encoding effort that the sending node needs to provide for combating the losses occurring from the non-simultaneous failures of links in the communication footprint. We show that by capillarization of the routing topology, we can substantially decrease the overall amount of FEC codes needed for reliable streaming.

7. References

[1] Amin Shokrollahi, “Raptor codes”, ISIT’04, June 27 – July 2, page 36

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

[3] Technical Specification Group Services and System Aspects, “Report of FEC selection for MBMS”, Mar 14-17, 2005, http://www.3gpp.org

[4] Technical Specification Group Services and System Aspects, “Efficient FEC code for reliable MBMS user services”, Mar 14-17, 2005, http://www.3gpp.org

[5] Technical Specification Group Services and System Aspects, “Report of MBMS FEC Status”, Jun 6-9, 2005, http://www.3gpp.org

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

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

[8] 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

[9] 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

[10] 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”, IEEE GLOBECOM’04 Vol. 2, Nov 29 - Dec 3, pp. 977-981

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

[12] Rui Ma, Jacek Ilow, “Reliable multipath routing with fixed delays in MANET using regenerating nodes”, LCN’03, Oct 20-24, pp. 719-725

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

[14] Thinh Nguyen, Avideh Zakhor, “Protocols for distributed video streaming”, Image Processing 2002, Vol. 3, Jun 24-28, pp. 185-188

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

[16] Seong-ryong Kang, Dmitri Loguinov, “Impact of FEC overhead on scalable video streaming”, NOSSDAV’05, Jun 12-14, pp. 123-128

[17] Youshi Xu, Tingting Zhang, “An adaptive redundancy technique for wireless indoor multicasting”, ISCC 2000, Jul 3-6, pp. 607-614

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

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

[20] John W. Byers, Michael Luby, Michale Mitzenmacher, “Accessing multiple mirror sites in parallel: using Tornado codes to speed up downloads”, INFOCOM 1999, Vol. 1, Mar 21-25, pp. 275-283

[21] 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