Capillary routing for reliable real-time streaming

Emin Gabrielyan, Roger D. Hersch
École Polytechnique Fédérale de Lausanne (EPFL)
Switzerland
{emin.gabrielyan, rd.hersch}@epfl.ch

Abstract

Application of forward error correction (FEC) in off-line streaming with large buffering time dramatically improves the quality and performance of communications in challenging network conditions. However real-time streaming puts hard restrictions on the buffer size leaving FEC helpless for combating long link failures on a single path route. Multi-path routing is another method, orthogonal to buffering, which can make FEC effective also for real-time streaming. In this paper we introduce a capillary routing algorithm offering layer by layer a wide range of multi-path routing topologies starting from a simple multi-path solution toward more reliable and secure schemes obtained by recursively spreading individual sub-flows. The friendliness of a particular multi-path routing scheme is rated by a measure called Failure Recovery Redundancy Overall Requirement (FRROR), which is proportional to the total amount of FEC codes required for combating the individual failures of all links in the multi-path route. A dozen of capillary routing layers, built on several hundreds of network samples obtained from a random walk wireless Mobile Ad-Hoc Network (MANET), are rated with FRROR. We show that the overall requirement in FEC decreases substantially as the spreading of the routing grows.

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.

In off-line packetized applications forward error correction (FEC) yields spectacular results. Via satellite broadcast channel with erasure resilient Raptor codes [Shokrollahi04] it is possible to recurrently update voluminous GPS maps to millions of motor vehicles under conditions of arbitrary fragmental visibility and without a feed-back channel [honda04]. In the film industry, LT codes [Luby02] 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) [hollywood03]. 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 [MBMS05a], [MBMS05b], [MBMS05c].

The above examples of off-line streaming can significantly benefit from FEC due to the fact that in 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 [Johansson02] improvements from the application of FEC result only if the stochastic packet losses range is between 1% and 5%. For real-time packetized streaming, in [Huang05] it is proposed to combine FEC with retransmissions. In [Padhye00] a high overhead has been reported from the use of FEC during bursts. The author of [Altman01] 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 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 [Qu04], [Tawan04], [Ma03], [Ma04], [Nguyen02] and [Nguyen03]. However these studies are limited to either two (possibly correlated) paths or in the best case to a sequence of parallel and serial links. Sets of differing routing topologies, so far, were not considered as search spaces for FEC effective routing patterns.

In this paper we present a comparative study of various multi-path routing schemes. Single path routing, being considered as too hostile, is excluded from our comparisons. In order to create steadily diversifying multi-path routing schemes, we propose *capillary routing*. In capillary routing the routing suggestions are proposed layer by layer and the path diversity develops as the layer number increases. Construction algorithms for capillary routing are described in sections 4 and 5.

In order to compare multi-path routing variants, we introduce a measure relying on a virtual application employing FEC and on its sender's transmission rate increases in response to individual link failures. Adjustable FEC for real-time streaming was proposed by several authors [Kang05], [Xu00], [Padhye00], [Johansson02] and [Huang05]. The end-to-end adaptive FEC mechanism is implemented entirely at the application level at the end nodes and is not aware of the underlying routing scheme. By default the sender is streaming the media with a constant FEC in order to tolerate a certain packet loss rate. The packet loss rate is measured at the receiver and is constantly reported back to the sender. 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 in respect to FEC 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. This value represents also a meaningful quantity, since relatively to the usual source packet transmission rate and the average failure time of a single network link, FRROR is simply the factor of the overall number of redundant packets that would be needed during the communication session. FRROR is presented in section 3.

In section 6, we evaluate the FEC friendliness 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 for multi-path routing, we can improve the efficiency of FEC by developing the basic path diversity provided by the first layer of the capillary routing (e.g. the max-flow solution) towards more elaborated multi-path routing strategies provided by the deeper layers of the capillary routing. Multi-path routing, similarly to buffering, can substantially burst the efficiency of FEC.

2. Cost of multi-path routing for real-time streaming

Multi-path routing is not necessarily wasteful in terms of overall network capacity (see Fig. 1 and Fig. 2.).

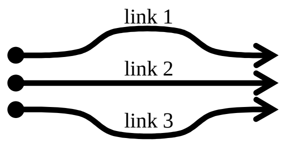

Fig. 1. Three transmissions following each a single path

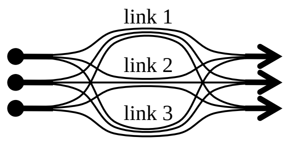

Fig. 2. Three transmissions of Fig. 1, each spread over multiple paths do not require extra network capacity

An increase in consumed network capacity may however occur when the multi-path routing footprint contains many alternate paths which are substantially longer than the shortest path. In terms of the overall network performance, the individual strategy of choosing the shortest path is often asymptotically efficient [Gafni87] and [Yang01]. However the shortest path approach pursues no support for achieving real-time media fault-tolerance. In the present paper we study multi-path routing in order to minimize the impact of link failures without dealing with the problem of overall network performance.

In a bursty and lossy public network where the traffic share by a streaming media, such as Voice over IP (VOIP), is tiny and negligible, even doubling the capacity share can be justified, if pursuing the fault-tolerance of streaming media. It is especially true, since, despite all Quality of Service (QoS) specifications, most of the time in public networks a tiny 18 kbps stream of VOIP packets is treated at the same priority level as TCP bursts, each one consuming a capacity of hundreds of kilobits/s.

Even for the portions of network carrying large shares of real-time media, a relative comparison of costs of network bandwidth and real-time communication justifies the fault-tolerance obtained at the cost of network's overall capacity. For example, an Internet Telephony Service Provider (ITSP), transiting through its core network large numbers of simultaneous real-time media sessions would lease additional capacities at the typical cost of about 300USD per 1Mbps per month in order to improve the quality and stability of phone conversations costing several tens of US cents, corresponding, in respect to 1Mbps bandwidth, to a factor of many tens or hundreds of thousands of USD per month.

In capillary routing, the alternate paths are discovered by delegating the load of a single path route to other links. Capillary routing is built layer by layer, providing at each layer a multi-path routing suggestion spreading out the traffic 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 out the individual sub-flows of previous layers. With the capillarization of the routing, the communication uses the network more reliably exploiting all transmission capacities offered by the network topology. The last layer represents the complete capillary routing and, in contrast to shortest path or max-flow solutions, has only one unique solution for any given network and pair of peers. We present the capillary routing construction algorithm in sections 4 and 5.

3. Failure Recovery Redundancy Overall Requirement (FRROR)

Most real-time media streaming applications are tolerant to a certain level of packet losses due to passive error concealment or media encoding techniques (such as carrying in a packet duplicates of media from previous slots, encoded with a lower rate source coding). 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 $5 \cdot 25\% = 1.25$.

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 we accordingly define:

$$FRROR = \sum_{l \in L} \left(\frac{P_l}{P} - 1 \right) \tag{1}$$

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. D is the product of the average duration of a single link failure, the frequency of a single link failure and the total communication time. $D \cdot P$ refers to the quantity of packets, which the sender would transmit at its usual rate P within the total time of all consecutive failures of one single network link. Then, according to equation (1), assuming a single link failure at a time and a uniform probability and duration of link failures, $D \cdot P \cdot ARON$ is the number of extra redundant packets that the sender actually needs to transmit in order to compensate for all network failures occurring during the total communication time. If for various routing suggestions parameters D and P do not change, FRROR is then the component which depends on the size and topology of a specific multi-path routing scheme. In other words, in respect to the total number of extra redundant packets i.e. in respect to the cost of redundancy in the communication, FRROR is the coefficient characterizing the FEC friendliness of a specific routing scheme.

Redundant packets are injected in the original media stream for every block of *M* media packets using systematic erasure resilient codes (thus without affecting the original media packets). The number of media packets (*M*) per transmission block is limited by the receiver's playback buffer time. During streaming, *M* is supposed to stay constant. The number of redundant packets for each block 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. Let us denote by FEC_p 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 $0 \le t < 1$ 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 failure (or congestion) time 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 $0 \le r(l) \le 1$ will produce at the receiver a random packet loss rate equal to the same relative traffic load $r(l)$. The equation (1) for FRROR can thus be re-written as follows:

$$FRROR = \underbrace{\sum_{l \in L \,|\, t \le r(l) < 1} \left(\frac{FEC_{r(l)}}{FEC_t} - 1 \right)}_{\text{a sum over all links carrying a flow exceeding the tolerable loss limit except the links passing the entire traffic}} \tag{2}$$

The links carrying the entire traffic are skipped in the sum index of equation (2), since the FEC required for the compensation of failures of such links is infinite. If the link is critical by the network topology, any routing suggestion will unavoidably pass its entire traffic through such a link, and therefore without a risk of affecting the comparison, the corresponding "equivalent" infinite components can be removed from the FRROR rates of all suggested routings in order to compare their remaining portions. We are assuming that none of considered multi-path routing schemes may pass its entire traffic through a non-critical single link.

We compute the FEC_p function with 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 of packets (of any type: media or redundant) as there are original media packets in the FEC block.

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

In order to collect a mean of *M* packets at the receiver, $M/(1-p)$ packets must be transmitted at the sender. However the probability of receiving $M-1$ packets or $M-2$ packets (which makes the decoding impossible) remains high. Therefore for transmitting blocks carrying a small number (*M*) of media packets, we must send much more redundant packets in the block than is necessary to receive an average of *M* packets at the receiver side. In order to maintain a very low probability δ of receiving less than *M* packets ($\delta \leq DER$) the average number of received packets should be much higher than *M*.

The probability of having exactly *n* losses (erasures) in a block of *N* packets with a random loss probability *p* is computed according to the binomial distribution:

$\binom{N}{n} \cdot p^n \cdot q^{N-n}$, where $\binom{N}{n} = \frac{N!}{n! \cdot (N-n)!}$ and $q = p-1$

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

$$\delta = \sum_{n=N-M+1}^{N} \binom{N}{n} \cdot p^n \cdot q^{N-n} \qquad (3)$$

The above formula gives the decoding failure probability if the FEC block size is equal to *N*. Therefore for computing the carrier block's minimal 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. We pre-compute the FEC block size as a function of the random loss probability $0 \leq p < 1$.

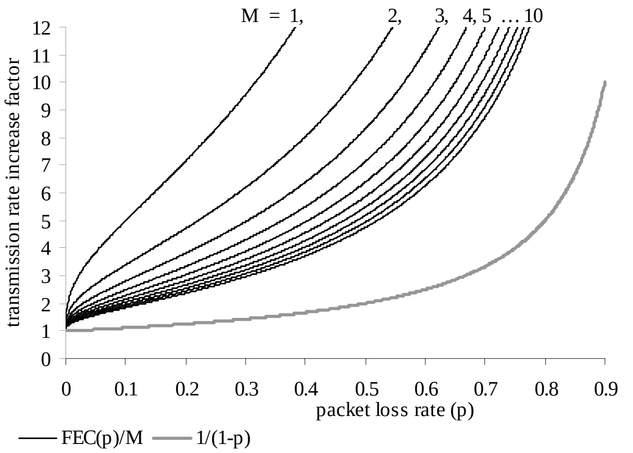

Fig. 3. FEC overhead as a function from the packet loss rate ($DER = 10^{-5}$)

Several transmission rate increase factors (FEC_p / M) for media blocks of size *M* from 1 to 10 are plotted in Fig. 3 (for $DER = 10^{-5}$). The transmission rate increase functions FEC_p / M of Fig. 3 are compared with a rate $1/(1-p)$, corresponding to the lowest theoretically possible transmission rate increase. The higher is the number of media packets in the block (i.e. longer is the buffering time) the closer the communication can approach the theoretical limit.

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).

In off-line applications the number of source packets *M* per transmission block can be maintained very large. In file transfers for example *M* can be the number of all packets in the entire file, or in one-way video the buffering time can be a few minutes long, with thousands of media packets within each single FEC block. In contrast to real-time media, when the number of packets in the transmission block is very large, for a given probability *p* of packet losses, the proportion of actually received packets remains very close to $1-p$. Although for very large numbers of source packets MDS codes do not exist, several other capacity approaching codes, such as erasure resilient fountain codes [MacKay05], can practically reach the theoretical limit. A packet loss rate *p* can be thus compensated by an increase of the encoded stream transmission rate only by the theoretical factor of $1/(1-p)$.

Path diversity can be required also in off-line streaming applications or in long downloads, e.g. for avoiding the idle times of the last kilometer bottleneck occurring during arbitrary failures in the lossy Internet. Thanks to the sender's adaptive transmission rate and to multi-path routing, the feeding of the last kilometer bottleneck link can be constantly maintained at its maximal bandwidth (see [Nguyen02] and [Byers99] for video streaming from multiple servers). Since the buffering is not a concern the application may use a capacity approaching fountain code, such that each individual network failure causing a packet loss rate p can be compensated by an increase of the transmission rate only by a theoretical factor of $1/(1-p)$, and with a good choice of multi-path routing the impact of individual network failures is minimal. The friendliness of the multi-path routing can be evaluated with FRROR. With the near optimal conditions $FEC_t = M/(1-t)$ and $FEC_{r(l)} = M/\left(1-r(l)\right)$, we derive from equation (2) the following relationship:

$$FRROR = \sum_{l \in L \mid t \le r(l) \triangleleft 1} \left(\frac{1-t}{1-r(l)} - 1 \right) \qquad (4)$$

The next two sections present algorithms for creating multi-path capillary routing schemes. Section 6 presents FRROR ratings of various capillary routing suggestions for real-time streaming.

4. Capillary routing

Capillary routing seeks to minimize the impact of individual link failures on real-time streaming, requiring 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.

Fig. 4, Fig. 5 and Fig. 6 show three layers of the capillary routing on a network example with 7 nodes. The top node on the diagrams is the sender and the bottom node is the receiver. All links are oriented from top to bottom.

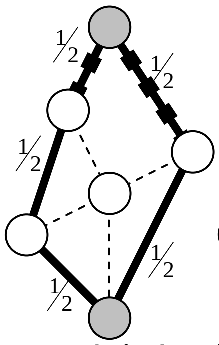

Fig. 4. In the first layer the flow is equally split across two paths, where two of their links marked by thick dashes are the bottlenecks.

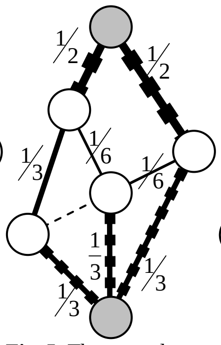

Fig. 5. The second layer minimizes the maximal load of the remaining seven links and finds three bottlenecks, each with a load of 1/3.

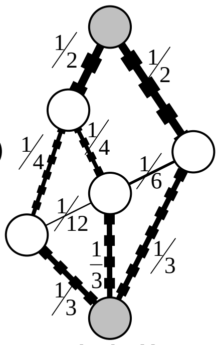

Fig. 6. The third layer minimizes the maximal load of the remaining four links and finds two bottlenecks, each with a load of 1/4.

In the first layer of the capillary routing (Fig. 4) there are four links having a load of $1/2$, but only two of them, marked by thick dashes, are true bottlenecks and will continue to maintain their load in all successive iterations. In the second layer (Fig. 5) there are four links with a value of $1/3$, but only three of them, also marked by thick dashes, are the true bottlenecks. In the third final layer (Fig. 6) there are two bottlenecks with a load of $1/4$. The loads of the two remaining links ultimately become $1/6$ and $1/12$ respectively.

Although the above described iterative LP process defining the capillary routing is completely valid for formulation of LP models, 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, can 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. 7 to Fig. 12 present the discovery of capillary routing according to the max-flow LP approach, on the same network example with the 7 nodes previously shown in Fig. 4, Fig. 5 and Fig. 6.

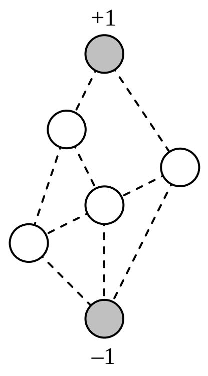

Fig. 7. Initial problem with one source and one sink node

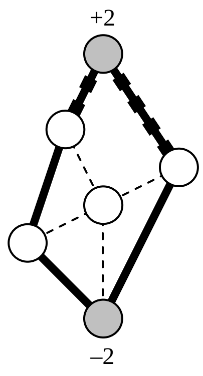

Fig. 8. Maximize the flow, fix the new flow-out coefficients at the nodes and find the bottleneck links

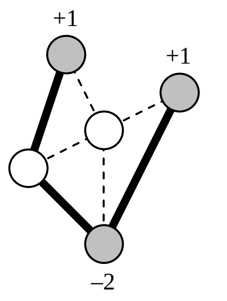

Fig. 9. Remove the bottleneck links from the network and adjust the flow-out coefficients at the adjacent nodes

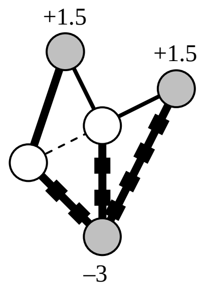

Fig. 10. Maximize the flow in the new sub-problem, fix the new flow-out coefficients at the nodes and find the new bottlenecks

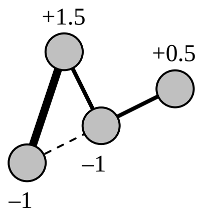

Fig. 11. Remove the bottleneck links from the network and adjust correspondingly the flow-out coefficients at the adjacent nodes

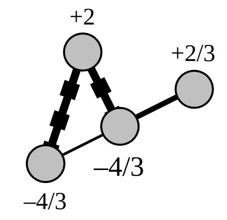

Fig. 12. Again maximize the flow in the obtained new problem and fix the new resulting flow-out coefficients at the nodes

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-multicast 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. 7 to Fig. 12) the flow increase factor of the network at the first layer (F^1) is equal to 2 (Fig. 8), the network flow increase factor at the second layer (F^2) is equal to 1.5 (from Fig. 9 to Fig. 10) and the flow increase factor of the third layer (F^3) is 4/3 (from Fig. 11 to Fig. 12).

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" [Fourer03].

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.

In the next section we describe the identification of bottlenecks after the max-flow solution of the capillary routing layer is found.

5. 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 of Fig. 13. The flow in 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. 14. 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*}.

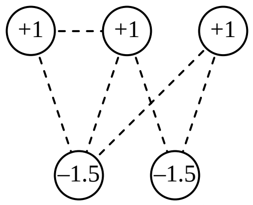

Fig. 13. An example of a bounded multiple-sources/multiple-sinks problem (obtained during construction of the capillary routing from a network with one source and one destination nodes)

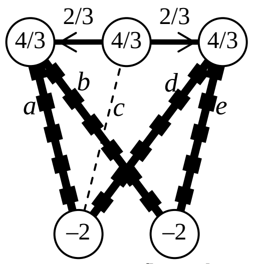

Fig. 14. A max-flow solution with the flow increase factor of 4/3, containing four maximally loaded candidate links {*a*, *b*, *d*, *e*}

The initial value of this total cost is equal to 4 in Fig. 14. The minimal value it can be reduced to is 3, and a possible solution is shown in Fig. 15. 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.

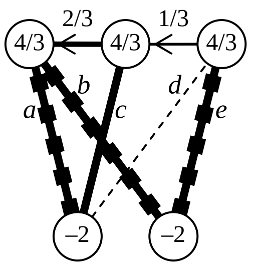

Fig. 15. Cost reduction applied to four fully loaded links of Fig. 14 reduced the load of a candidate link *d*, and the candidate links are now {*a*, *b*, *e*}.

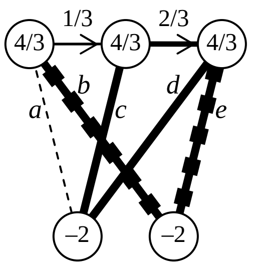

Fig. 16. Cost reduction applied to the three fully loaded links of Fig. 15 reduced the load of another candidate link *a*, and the true bottleneck links are {*b*, *e*}.

By minimizing the 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. 16). 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. 14 and Fig. 16, link *c* in Fig. 15 and Fig. 16), but they are not bottlenecks, since there were always 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. 17 shows an example of capillary routing, built simultaneously on 200 unbounded networks with 300 nodes in each (total 60,000 nodes and 538,940 links). We show the decrease in the average number of suspected links as the bottleneck hunting of a given capillary routing layer progresses. The bottleneck hunting of the presented example requires from 10 to 25 iterations, depending on the layer of the capillary routing (from 1 to 10).

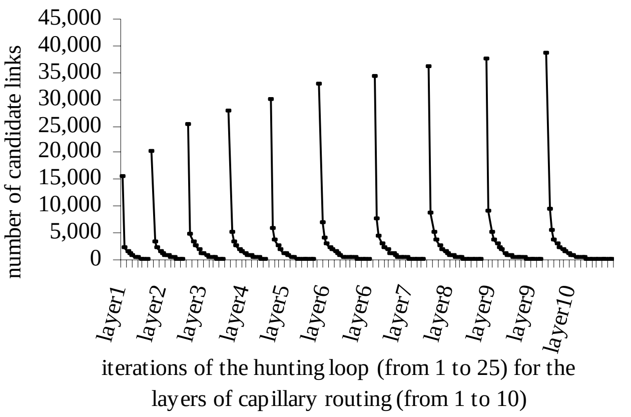

Fig. 17. Evolution of the number of candidate links during bottleneck hunting loops, for 10 capillary routing layers on a problem with 200 unbounded networks

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

6. Measuring the 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. 18, 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.

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 however reduces considerably the total FRROR rating and therefore also the FEC effort of the sender combating the link failures and packet losses.

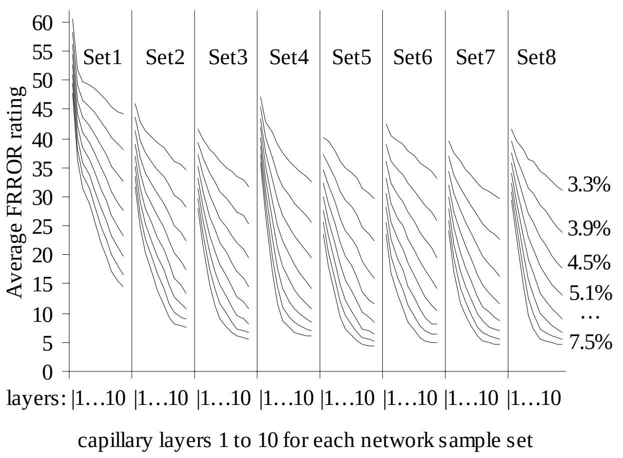

Fig. 18. Average FRROR as a function from the capillary routing layer (the static tolerance of the stream from 3.3%, for the upper curves, to 7.8%, for the lower curves, by a step of 0.6%)

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. Spread routing alone would not solve the problem of tolerance without FEC. If a media stream is not capable to sustain any loss at all, by spreading the transmission, the media becomes even more vulnerable, since there are more links whose failures will damage the stream.

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. According to the chart, streams tolerating a 6% or higher packet loss rate almost do not gain from spreading beyond layer 8.

The exact pattern of the FRROR improvement 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 number of media packets (M) per transmission block is 20 and the desired decoding failure rate (DER) is 10^{-5}.

7. Applications of capillary routing

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 accessed, path diversity can be still obtained in upper network or application layers.

Spreading in IP networks (without having access to the physical routing) can be achieved transparently by splitting the flow at the source across VPN tunnel interfaces leading to several VPN gateways scattered across the network. Alternatively, path diversity can be also obtained by assigning to media gateways several IP interfaces connected to networks of different Internet Service Provider (ISP). More flexible method relies on overlay networks over public Internet using peer-to-peer relay nodes (see [Nguyen03] and [Guven04]). For example, the relay nodes of an ITSP shall simply forward the UDP packets without processing their content. Spreading of the flow from the end user to the relay nodes can be implemented at the closest Service Initiated Protocol (SIP) proxy server, in the firmware of a SIP phone or in the Network Address Translation (NAT) router of the user (dynamically translating the target IP address of the outgoing UDP packets to the IP addresses of intermediate media routers).

Modifying the physical routing by ISP requires the least overhead. Most of the UDP packets carry streaming media, and since capillary routing is good for any type of real-time or off-line media streaming, irrespectively to the specific media, an ISP can simply spread out the routing of the entire mix of UDP packets traversing its network. The ISP needs to properly balance at each router the outgoing traffic across its outgoing interfaces. Most IP routers, including all recent IOS releases of Cisco gateways, provide load balancing in static routing mode or in Enhanced Interior Gateway Routing Protocol (EIGRP) mode (used in the packet load balance mode instead of the typical session load balance mode).

8. 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 as the routing capillarizes, we show a substantial improvement of the routing topology and increase of its FEC efficiency.

For special networks exclusively dedicated to media streaming and when the network overall capacity may be really an issue (e.g. in a LEO satellite network similar to the Iridium network), one may try to further optimize the routing, relying on the capillary routing footprints of each individual communication session and re-distributing the flow within each footprint trying to optimize also the overall usage of network capacities. Since there exist only one solution of capillary routing for any given network and pair of communicating nodes, capillary routing pattern is a good starting point.

9. References

[Shokrollahi04] Amin Shokrollahi, “Raptor codes”, ISIT’04, June 27 – July 2, page 36

[Luby02] Michael Luby, “LT codes”, FOCS’02, November 16-19, pp. 271-280

[MBMS05a] Technical Specification Group Services and System Aspects, “Report of FEC selection for MBMS”, Mar 14-17, 2005, http://www.3gpp.org

[MBMS05b] Technical Specification Group Services and System Aspects, “Efficient FEC code for reliable MBMS user services”, Mar 14-17, 2005, http://www.3gpp.org

[MBMS05c] Technical Specification Group Services and System Aspects, “Report of MBMS FEC Status”, Jun 6-9, 2005, http://www.3gpp.org

[Johansson02] Ingemar Johansson, Tomas Frankkila, Per Synnergren, “Bandwidth efficient AMR operation for VoIP”, Speech Coding 2002, Oct 6-9, pp. 150-152

[Huang05] Yicheng Huang, Jari Korhonen, Ye Wang, “Optimization of Source and Channel Coding for Voice Over IP”, ICME’05, Jul 06, pp. 173-176

[hollywood03] Mark Fritz, “Digital Dailies Flow Freely from Fountain”, April 1, 2003, http://www.emedialive.com/Articles/ReadArticle.aspx?CategoryID=45&ArticleID=5077

[honda04] Loring Wirbel, “Deal pushes algorithms into digital radio”, April 13, 2004, http://www.commsdesign.com/showArticle.jhtml?articleID=18901216

[Padhye00] 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

[Altman01] 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

[Qu04] 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

[Tawan04] Tawan Thongpook, “Load balancing of adaptive zone routing in ad hoc networks”, TENCON 2004, Vol. B, Nov 21-24, pp. 672-675

[Ma03] Rui Ma, Jacek Ilow, “Reliable multipath routing with fixed delays in MANET using regenerating nodes”, LCN’03, Oct 20-24, pp. 719-725

[Ma04] Rui Ma, Jacek Ilow, “Regenerating nodes for real-time transmissions in multi-hop wireless networks”, LCN’04, Nov 16-18, pp. 378-384

[Nguyen02] Thinh Nguyen, Avideh Zakhor, “Protocols for distributed video streaming”, Image Processing 2002, Vol. 3, Jun 24-28, pp. 185-188

[Nguyen03] Thinh Nguyen, P. Mehra, Avideh Zakhor, “Path diversity and bandwidth allocation for multimedia streaming”, ICME’03 Vol. 1, Jul 6-9, pp. 663-672

[Kang05] Seong-ryong Kang, Dmitri Loguinov, “Impact of FEC overhead on scalable video streaming”, NOSSDAV’05, Jun 12-14, pp. 123-128

[Xu00] Youshi Xu, Tingting Zhang, “An adaptive redundancy technique for wireless indoor multicasting”, ISCC 2000, Jul 3-6, pp. 607-614

[Fourer03] Robert Fourer, *A modeling language for mathematical programming*, Thomson – Brooks/Cole, second edition, 2003, page 343

[MacKay05] David J. C. MacKay, “Fountain codes”, IEE Communications, Vol. 152 Issue 6, Dec 2005, pp. 1062-1068

[Byers99] 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

[Guven04] 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

[Gafni87] Eli M. Gafni, Dimitri P. Bertsekas, “Asymptotic optimality of shortest path routing algorithms”, Information Theory 1987, Vol. 33, Iss. 1, Jan, pp. 83-90

[Yang01] Shanchieh Yang, Xun Su, Gustavo De Veciana, “Heterogeneity-aware shortest path routing: flow holding time, user demand and network state”, HPSR’01, May 29-31, pp. 287-291