# IMPLEMENTATION AND EVOLUTION OF PACKET STRIPING FOR MEDIA STREAMING OVER MULTIPLE BURST-LOSS CHANNELS

*Gene Cheung, Puneet Sharma, Sung-Ju Lee*

Hewlett-Packard Laboratories

**ABSTRACT**

Modern mobile devices are multi-homed with WLAN and WWAN communication interfaces. In a community of nodes with such multi-homed devices — locally inter-connected via high-speed WLAN but each globally connected to larger networks via low-speed WWAN, striping high-volume traffic from remote large networks over a bundle of low speed WWAN links can overcome the bandwidth mismatch problem between WLAN and WWAN. In our previous work, we showed that a packet striping system for such multi-homed devices — a mapping of delay-sensitive packets by an intermediate gateway to multiple channels using combination of retransmissions (ARQ) and forward error corrections (FEC) — can dramatically enhance the overall performance. In this paper, we improve upon a previous algorithm in two respects. First, by introducing two-tier dynamic programming tables to *memoize* computed solutions, packet striping decisions translate to simple table lookup operations given stationary network statistics. Doing so drastically reduces striping operation complexity. Second, new weighting functions are introduced into the hybrid ARQ/FEC algorithm to drive the long-term striping system evolution away from pathological local minima that are far from the global optimum. Results show the new algorithm performs efficiently and gives improved performance by avoiding local minima compared to the previous algorithm.

## 1. INTRODUCTION

Modern wireless devices are multi-homed — having multiple wireless communication interfaces, each connecting to the Internet via a wireless wide area network (WWAN) interface such as a cellular link, and connecting to a community of neighboring nodes via a wireless local network (WLAN) interface. Though WWAN provides long range services, the bandwidth is limited, and packet losses are frequent and bursty. To enhance performance in this setting, an assistant gateway can "aggregate" a community of devices' low speed WWAN channels — a mapping of incoming packets destined for a community member to the community's multiple channels — to optimize end-to-end packet delivery. The received packets are subsequently forwarded to the interested community member via high-speed WLAN connections [1, 2, 3].

Such multi-path *striping* engine have been investigated to optimize TCP flows [4], to minimize end-to-end delay of media traffic over lossless channels [5], and to increase robustness of multiple description video over failure-prone channels [6]. Less obvious is that striping engine greatly improves delivery of delay-sensitive media streaming data over burst-loss channels by striping forward error correction (FEC) [7] and retransmissions (ARQ). For FEC, similar to a single channel packet interleaver, striping spreads FEC packets across channels and avoids decoding failure due to a single burst loss. Yet unlike the interleaver, striping also avoids excessive transmission delay of long interleaving in a lone channel. We call this the *interleaving effect*. For ARQ, given a packet's delivery deadline, striping judiciously selects the best performing channel that optimizes a packet's survival — one that maximizes its successful transmission probability *and* its chance for retransmission if the current transmission fails. We call this the *transmission / retransmission bundle*.

In previous work [8, 9], we showed that a striping engine can indeed greatly improve the delivery of delay-sensitive packets, where [8] derives striping algorithms for burst-loss channels with constant delays and [9] derives for burst-loss channels with random delays. However, two inherent problems remain, preventing the striping algorithms from wide adoption. One, the complexity of the striping algorithms — the hybrid ARQ/FEC algorithm in [9] in particular — is high and hence impractical to be performed packet-by-packet. Two, on occasions the hybrid ARQ/FEC algorithm, optimizing delivery on a per-packet basis, can drive the striping system evolution to pathological local minima far from the globally optimum.

In this paper, we present a new algorithm that specifically addresses these two concerns. In Section 2, we overview the striping system, models and algorithms that were detailed in [9]. In Section 3, we discuss a two-tier implementation of dynamic programming tables that drastically speeds up the execution of the striping algorithm given stationary network statistics. In Section 4, we introduce weighting functions into the hybrid FEC/ARQ algorithms to drive the long-term striping system evolution away from pathological local minima that are far from global optimum. Results and conclusions are presented in Section 5 and 6, respectively.

## 2. OVERVIEW OF PACKET STRIPING SYSTEM

We overview the packet striping system in [9]. We first describe the network model, then outline the ARQ-based, the FEC-based, and the hybrid ARQ/FEC algorithm in order.

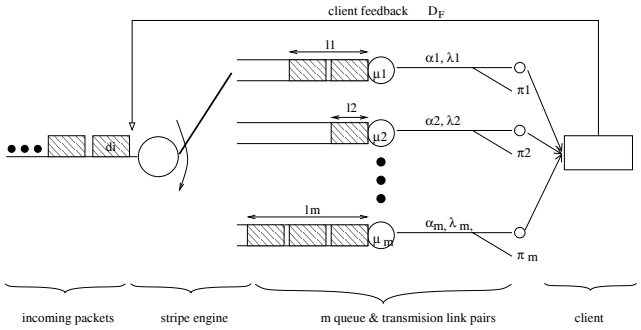



**Fig. 1**. Bandwidth-limited Network Model

### 2.1. Network Model

The network model we adopted is a bandwidth-limited, burst-loss model with random delays as shown in Figure 1. Each $j$ of $m$ channels is modeled by a FIFO queue and transmission link pair: a queue with constant service rate $\mu_j$ is connected to a transmission link of shifted-Gamma-distributed random variable delay $\gamma_j \sim \mathcal{G}(\kappa_j, \alpha_j, \lambda_j)$ with probability distribution function (pdf) $g_{\Gamma_s}(\gamma)$, and Gilbert-modeled burst loss of parameters $p_j$ and $q_j$. At a given time, the fullness of the queue $j$ is $l_j$. The time required to transmit a packet through queue $j$ is then: $(l_j + 1)/\mu_j + \gamma_j$. The client can inform the striping engine of a loss event losslessly in constant time $D_F$.

Packets are injected into the incoming queue before the striping engine, each labeled with an expiration time $d_i$. A packet with $d_i$ must be delivered by time $d_i$ or it expires and becomes useless. The packets are ordered in the incoming queue by earliest expiration times. Striping engine is activated whenever there is a packet in the incoming queue.

### 2.2. ARQ-based Algorithm

We now outline the ARQ-based striping algorithm [9]. We choose to optimize one packet at a time, each with expiration time $d$. Let $f_{ARQ}(d'), d' = d - t$, be the probability that a packet with expiration $d$ is timely delivered to the client using (re)transmission (ARQ), where $t$ is the optimization instant. Let $f^{(i)}_{ARQ}(d')$ be the probability that the same packet is timely delivered if channel $i$ is first used for ARQ. If failed, the packet has a chance for retransmission with a tighter deadline. We can write:

$$f_{ARQ}(d') = \begin{cases} \max_{i=1,\ldots,m} f^{(i)}_{ARQ}(d') & \text{if } d' \geq 0 \\ 0 & \text{o.w.} \end{cases}$$

$$f^{(i)}_{ARQ}(d') = \int_{\kappa_i}^{d' - \left(\frac{l_i+1}{\mu_i}\right)} g_{\Gamma_s}(\gamma)\left((1-\pi_i) + \pi_i f(d' - D_F - \gamma)\right) d\gamma \quad (1)$$

where $\pi_i = p_i/(p_i + q_i)$ is the raw packet loss rate (PLR) of channel $i$. The reason for such integration interval is that $g_{\Gamma_s}(\gamma)$ is zero for transmission $\gamma < \kappa_i$, and the packet will miss its deadline $d$ for $\gamma > d' - \left(\frac{l_i+1}{\mu_i}\right)$.

As (1) is defined recursively within an integral, it is difficult to solve directly. Instead, we first approximate (1) using *quantization* before resolving the recursive calls. By quantization, we mean we divide the integration interval of $g_{\Gamma_s}(\gamma)$ into $L$ evenly spaced regions, where region $l$ has boundaries $[b^{(i)}_{l-1}, b^{(i)}_l)$. By construction then, transmission delays $\gamma$'s in each region $l$ are upper-bounded by boundary $b^{(i)}_l$. If we quantize all the delays in each region $l$ to $b^{(i)}_l$, each region has probability $\int_{b^{(i)}_{l-1}}^{b^{(i)}_l} g^{(i)}_{\Gamma_s}(\gamma)d\gamma$. We can now approximate (1) as:

$$f^{(i)}_{ARQ}(d') \approx \sum_{l=1}^{L} \int_{b^{(i)}_{l-1}}^{b^{(i)}_l} g^{(i)}_{\Gamma_s}(\gamma)d\gamma \left[(1-\pi_i) + \pi_i f\left(d' - D_F - b^{(i)}_l\right)\right] \quad (2)$$

Now (2) can be solved recursively.

### 2.3. FEC-based Algorithm

We now outline the FEC-based striping algorithm [9]. We define an *FEC distribution* $\mathbf{g}$ as a particular mapping of $n-k$ parity and $k$ data packets to a set of $m$ channels for a given Reed-Solomon (RS) $(n,k)$ code. We will assume a known greedy algorithm [9], growing an FEC distribution one packet at a time greedily, is used to find a sub-optimal but good FEC distribution $\mathbf{g}$ for a given RS $(n,k)$ to be deployed on a set of $m$ burst-loss channels. In general, we consider RS $(n,k)$ while varying $n$ and $k$ for different channel coding strengths and FEC encoding/decoding delays. Let $f_{FEC}(d'_1), d'_1 = d_1 - t$, be the probability that a packet with expiration $d_1$ is timely delivered using FEC. We lower-bound the decoding success probability $f^{\mathbf{g}}_{n,k}(d'_i)$ of each of $k$ packets with expiration time $d_i$ with the FEC decoding success probability of the first packet $f^{\mathbf{g}}_{n,k}(d'_1)$. We can now write $f_{FEC}(d'_1)$ as:

$$\begin{aligned} f_{FEC}(d'_1) &= \max_{(n,k)} \left[\frac{1}{k}\sum_{i=1}^{k} f^{\mathbf{g}}_{n,k}(d'_i) - \lambda\left(\frac{n-k}{k}\right)\right] \\ &\approx \max_{(n,k)} f^{\mathbf{g}}_{n,k}(d'_1) - \lambda\left(\frac{n-k}{k}\right) \end{aligned} \quad (3)$$

where $f_{FEC}(d'_1)$ is optimized over a range of $n$ and $k$. Derivation of decoding probability $f^{\mathbf{g}}_{n,k}(d'_1)$ given FEC distribution $\mathbf{g}$ for a set of $m$ burst-loss channels can be found in [9].

Notice there is a *penalty* term $\lambda(\frac{n-k}{k})$ in (3). The reason is that using RS $(n,k)$ invariably increases the traffic volume by $(n-k)/k$ fraction more parity packets. Hence $\lambda$, proportional to the current packet volume in the outgoing queues, is used to regulate the packet volume so that it does not lead to queue overflows. Appropriate $\lambda$ is selected using standard numerical analysis [9].

### 2.4. Hybrid ARQ/FEC Algorithm

We can combine the ARQ and FEC algorithms into one hybrid algorithm. $f(d'_1)$ is then simply the larger value of the two possible choices — (re)transmission or FEC:

$$f(d'_1) = \begin{cases} \max[\, f_{ARQ}(d'_1),\ f_{FEC}(d'_1)\,] & \text{if } d'_1 \geq 0 \\ 0 & \text{o.w.} \end{cases} \quad (4)$$

Unlike (3), the FEC decoding success probability given FEC distribution g, $f^{\mathbf{g}}_{n,k}(d'_1)$, now spawns $L$ recursive calls to $f()$ with tighter delivery deadlines, similarly done for the ARQ-based algorithm in Section 2.2, to permit retransmission (reFEC) at the striping engine if sufficient loss events arrive at the client to signify an FEC decoding failure. Detailed derivations are similarly done and hence omitted.

## 3. IMPLEMENTATION OF HYBRID ARQ/FEC

It is clear that the hybrid ARQ/FEC algorithm described in Section 2.4 is computation-intensive. To execute the algorithm for every incoming packet is not possible. In this section we propose a two-tier dynamic programming (DP) implementation so that when the network statistics remain stationary (parameters of the network model remain the same), packet striping becomes simple table lookup operations.

The first tier of DP is used when (4) is solved for the first time. Because (4) recursively calls $f()$ with smaller arguments repeatedly, computed value of $f(a)$ can be stored in the $a^{th}$ entry of dynamic programming (DP) table $F[\,]$, so that future recursive calls of same argument can be simply looked up rather than re-computed. Further, we can restrict the size of the DP table to a limit of $H$ entries, placing an upper bound on the execution time. To do so, we must derive an index $a'$ into the table by first dividing the argument $a$ of $f(a)$ by constant $K$ to place or retrieve a value into or from the table. $K \in \mathcal{R}$ can be selected so that all possible arguments $a$'s map just inside the available space $H$:

$$K = \frac{a_{\max}}{H-1} \qquad a' = \left\lfloor \frac{a}{K} \right\rfloor \quad (5)$$

where $a_{\max}$ is the largest possible argument for (4). Because $f()$ is monotonically non-decreasing by definition, the rounding down operation provides a lower bound when calculating $f()$ recursively using the table.

The second tier of dynamic programming is used when parameters of the network models remain unchanged from packet to packet. Observe that the algorithm is computed based only on lifetime $d'$ and sizes of outgoing queues; each time $f(d')$ is computed using (4), the solutions should be stored in entry $[d'][l_1][l_2][l_3]$ of a DP table $Soln$. When a future packet arrives with survival time $d'$ and observable queue sizes $l_1$, $l_2$ and $l_3$, the striping engine can have its solution simply looked up in $Soln$. Similar dividing and rounding operation by the same constant factor $K$ can be done as well to further reduce complexity at the cost of solution quality.

### 3.1. Complexity Analysis

We analyze the complexity of solving (4) for the first time using first-tier DP as follows. The complexity is bounded by the time needed to construct the DP table of size $H$. Each entry is computed using (4), a comparison of $f_{ARQ}(d'_1)$ and $f_{FEC}(d'_1)$. $f_{ARQ}(d'_1)$ is computed by comparing $m$ $f^{(i)}_{ARQ}(d'_1)$'s in (1), each composes of a summation of $L$ terms in (2). $F_{FEC}(d'_1)$ tries all feasible combinations of $n$ and $k$, $k < n \leq n_{\max}$ in (3). Assuming $f^{\mathbf{g}}_{n,k}(d'_1)$ can be computed in $\mathcal{O}(L)$, the complexity of (4) is:

$$\mathcal{O}(\left(mL + n^2_{\max} L\right) \frac{a_{\max}}{K}) = \mathcal{O}(L\, n^2_{\max}\, \frac{a_{\max}}{K}) \quad (6)$$

where we assume $m \leq n_{\max}$. In practice, $a_{\max}$ dominates other terms in (6). Hence a small $H$ (large $K$) can effectively control the computation of (4).

Analysis of the algorithmic complexity of using second tier DP when the network statistics are stationary is a natural extension and hence is omitted.

## 4. DRIVING HYBRID ARQ/FEC ALGORITHM

It should not be surprising that the hybrid ARQ/FEC algorithm described in Section 2.4 evolves to sub-optimal local minima on occasions; the fact remains that we are optimizing one packet at a time with little regard for outgoing queue evolution and subsequent long-term system performance. Obviously, optimizing the striping system evolution via exhaustive search through all possible evolution paths is infeasible.

Fortunately, the penalty function $\lambda\left(\frac{n-k}{k}\right)$ in FEC-based algorithm (3) in Section 2.3 offers a hint of how objective function should be altered in face of abnormal system behavior — like ballooning packet volume in outgoing queues, though it alone is insufficient to drive the system from all pathological local minima. Consider the following degenerate case observable from actual experiment. Three outgoing queues of three channels are of packet volume $\{2,1,0\}_t$ at time $t$. Channel 0 and 1 are of each of equal raw PLR, with channel 2 of higher PLR but smaller queuing delay $\mu_2$. Hybrid algorithm dictates that the next packet be placed in Channel 1, creating an evolution of $\{2,1+\mathbf{1},0\}_t$, $\{1,2,0\}_{t+1}$, $\{1+\mathbf{1},2,0\}_{t+1}$, $\{2,1,0\}_{t+2}$, ... etc. Yet the system optimum turns out to be placement of packet in Channel 2 at time $t$, creating a chance for RS(3,2) to be spread across three channels later in time: $\{2,1,0+\mathbf{1}\}_t$, $\{1,1,0\}_{t+1}$, $\{1+\mathbf{1},1+\mathbf{1},0+\mathbf{1}\}_{t+1}$.

### 4.1. Finding Weighting Functions for ARQ

From the degenerate example, we see that ARQ could drive the striping system away from any selection of FEC, resulting in a sub-optimal local minimum. To prevent this pathology, we introduce a weighting function for the ARQ recursion (2) of channel $i$, proportional to the current queue length $i$:

$$\hat{f}^{(i)}_{ARQ}(d') = f^{(i)}_{ARQ}(d') - l_i\, \rho_i \quad (7)$$

where the weighting parameter $\rho_i$ is determined as follows. We first determine a desirable FEC distribution $\mathbf{g} = (\mathbf{u}, \mathbf{v})$ that is deemed worthy of consideration — one on the convex hull of the PLR-$\left(\frac{n-k}{k}\right)$ FEC performance curve, and whose

**Table 1**. Model Parameters for the Experiments.

| chnl | $p$ | $q$ | $\mu$ | $\alpha$ | $\lambda$ | $\kappa$ |
|---|---|---|---|---|---|---|
| 1 | 0.05 | 0.45 | $30ms/pkt$ | 4 | 0.2 | 50 |
| 2 | 0.03 | 0.27 | $30ms/pkt$ | 4 | 0.2 | 50 |
| 3 | 0.05 | 0.4 | $25ms/pkt$ | 4 | 0.16 | 50 |

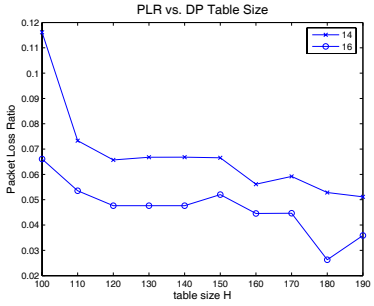



**Fig. 2**. Complexity / Performance Tradeoff.

fraction of increased parity $(n-k)/k$ does not overwhelm the outgoing queues given input packet volume. We then increase the outgoing queues one packet at a time using ARQ greedily until $\mathbf{g}$ is inferior in PLR compare to ARQs. Given this set of queue lengths, $l_i'$'s, we then determine the channel with the worst PLR ($\pi_{\max}$). Weighting parameter $\rho_i$ is then:

$$\rho_i = \frac{\pi_{\max} - \pi_i}{l_i'} \tag{8}$$

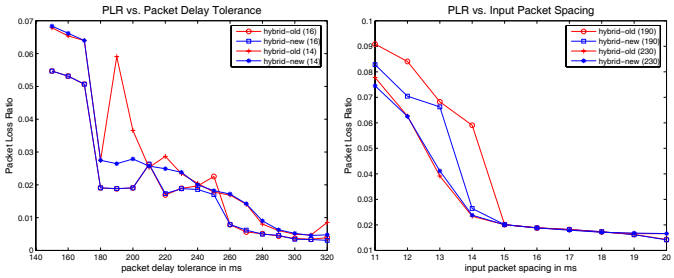
Doing so will ensure that when queue length reaches $l_i'$ for channel $i$, the algorithm will select the worst channel before channel $i$ for ARQ before completely eliminating FEC distribution $\mathbf{g}$ from consideration, driving future queue evolution to cases that eliminate $\mathbf{g}$ all together.

## 5. RESULTS

We constructed a network simulator called `muns` (**MU**lti-path **N**etwork **S**imulator) in `C` running on `Linux`. Network model parameters used are shown in Table 1. For each PLR data point, $500,000$ packets were inputed for an averaging effect.

We first investigate the tradeoff of complexity reduction against performance as discussed in Section 3. With end-to-end packet delay tolerance $a_{\max}$ fixed at $200ms$, in Figure 2 we plotted PLR against the first-tier DP table size $H$, for two trials of input packet spacing of $14ms$ and $16ms$ respectively. We see that PLR is generally inversely proportional to $H$, and that reasonable PLR can be achieved even for small $H$.

To test the effectiveness of the weighting functions for ARQ (7) in preventing the striping system from evolving to sub-optimal local minima, we compare the performance of the modified hybrid algorithm `hybrid-new` with the ARQ weighting functions with the original version `hybrid-old` in Figure 3. In Figure 3 (a) we plotted PLR against end-to-end




(a) PLR vs. Packet Delay Tolerance. (b) PLR vs. Input Packet Spacing.

**Fig. 3**. Performance of Different Hybrid ARQ/FEC.

packet delay tolerance in msec for two trials of input packet spacing of $16ms$ and $14ms$ respectively. In Figure 3 (b), we plotted PLR against input packet spacing in msec, for two trials of fixed end-to-end packet delay tolerance of $190ms$ and $230ms$ respectively. We see that `hybrid-new` performed as least as well as `hybrid-old`, and in cases where `hybrid-old` performed very poorly due to evolution to local minima, `hybrid-new` avoided the pitfalls and performed much better. This proves experimentally the effectiveness of the ARQ weighting functions.

## 6. CONCLUSION

In this paper, we addressed two inherent problems with the previous hybrid ARQ/FEC striping algorithm: i) by using a two-tier implementation of dynamic programming tables, execution of the algorithm for stationary network statistics is sped up; and, ii) by introducing weighting functions for ARQ, the striping system is driving away from poor-performing local minima far from global optimum.

## 7. REFERENCES


[1] A. C. Snoeren, "Adaptive inverse multiplexing for wide area wireless networks," in *Proceedings of the IEEE GLOBECOM'99*, 1999.

[2] H.-Y. Hsieh and R. Sivakumar, "A transport layer approach for achieving aggregate bandwidth on mutli-homed mobile hosts," in *Proceedings of the ACM MobiCom 2002*, Atlanta, GA, Sept. 2002, pp. 83–94.

[3] P. Sharma, S.-J. Lee, J. Brassil, and K. Shin, "Distributed communication paradigm for wireless community networks," in *IEEE International Conference on Communications*, Seoul, Korea, May 2005.

[4] C. Cetinkaya and E.W. Knightly, "Opportunistic traffic scheduling over multiple network paths," in *IEEE INFOCOM*, Hong Kong, March 2004.

[5] S. Mao, S. Panwar, and Y. Hou, "On optimal traffic partitioning for multipath transport," in *IEEE INFOCOM 2005*, Miami, FL, March 2005.

[6] S. Mao, Y.T. Hou, X. Cheng, H.D. Sherali, and S.F. Midkiff, "Multipath routing for multiple description video in wireless ad hoc networks," in *IEEE INFOCOM 2005*, Miami, FL, March 2005.

[7] T. Nguyen and A. Zakhor, "Path diversity with forward error correction (PDF) system for packet switched networks," in *IEEE INFOCOM*, San Francisco, CA, April 2003.

[8] G. Cheung, P. Sharma, and S. J. Lee, "Striping delay-sensitive packets over multiple bursty wireless channels," in *IEEE International Conference on Multimedia and Expo*, Amsterdam, the Netherlands, July 2005.

[9] G. Cheung, P. Sharma, and S. J. Lee, "Striping delay-sensitive packets over multiple burst-loss channels with random delays," in *IEEE International Symposium on Multimedia*, Irvine, CA, December 2005.