Low-Delay Dynamic Routing Using Fountain Codes

Venkataramana Badarla, Vijay Subramanian, and Douglas J. Leith

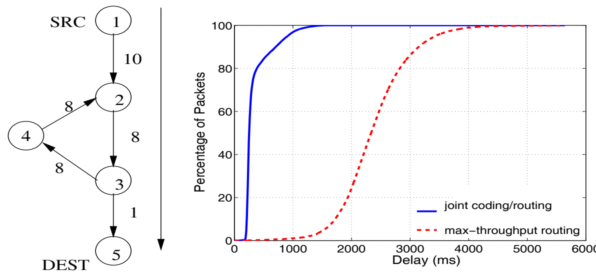
***Abstract*—This paper considers augmenting current maximum throughput routing algorithms to use fountain coding at senders. We demonstrate that this joint routing/coding approach is able to achieve significantly improved delay performance.**

***Index Terms*—Dynamic routing, fountain codes.**

I. Introduction

FOLLOWING the seminal work of Tassiulas [1], in recent years a body of powerful theoretical results has been developed for maximum throughput routing (e.g. see [2], [3] and references therein). Under quite mild conditions this yields distributed dynamic routing algorithms that are guaranteed to maximise network throughput. By using dynamic multi-path routing these algorithms can offer considerable gains in throughput over conventional single-path routing. Indeed, they are guaranteed to achieve the network capacity and so their throughput performance cannot be bettered by any algorithm. However, despite the appealing simplicity and strong theoretical basis for these routing algorithms, the literature is confined to analytic results with almost no simulation or experimental studies exploring their practical utility (except the recent work in [4]). In this paper we consider the application of maximum throughput routing algorithms and highlight some fundamental difficulties with current algorithms, in particular a tendency towards extensive routing loops and poor delay performance. Motivated by these observations, we propose augmenting current maximum throughput routing algorithms to use fountain coding at senders. We demonstrate that this joint routing/coding approach is able to achieve significantly improved delay performance. To our knowledge this paper is the first to consider the integrated use of sender-side coding with maximum throughput routing. We note briefly that we consider sender-side coding rather than network coding here – the reason being our focus on improving delay performance rather than increasing network capacity. Extending the proposed approach to include network coding is feasible but left as future work.

Before proceeding we look at an illustrative example. Consider the simple network topology in Fig. 1(a) where the network capacity between the source and destination is 1000 packet/s (constrained by the link connecting the destination node 5 with node 3). For Poisson arrivals, the delay performance of the maximum throughput routing (see Algorithm 1 which uses queue length differences between the adjacent nodes weighted by their link rates to make routing decisions) is shown in Fig. 1(b). We measure delay as the time between a packet being transmitted by the source and being delivered in-order to upper layers at the destination. It can be seen that 75% of packets require more than 2000ms to reach the destination. In general, while the routing algorithm is guaranteed to maximise network throughput this guarantee says little about the delay performance and, as can be seen from this example, in practice delay performance may be poor.



(a) Topology (b) Cumulative distribution of packet delay

Fig. 1. Example illustrating performance issues. Link rates in (a) marked in $\times 1000$ packets/s. Delay data in (b) is for Poisson arrivals, mean rate 900 packets/s, block size $n = 50$ packets.

Closer inspection reveals that packets tend to circulate for long periods in the loop connecting nodes 2-3-4. On average the number of packets exiting this loop from node 3 to the destination node 5 matches the arrival rate from source node 1 into node 2 and so network throughput is indeed maximised in line with analytic guarantees. Nevertheless, it is easy to see that up to 7000 packets/s can flow around the 2-3-4 loop without impacting the capacity between the source and destination. Consequently, the packet flow arriving at the destination can suffer from extensive packet re-ordering. Note that when in-order (or near in-order) packet delivery is required (as is the case for the vast majority of existing applications), packet reordering translates into delay as packets must be held in a reassembly queue (akin to a playout buffer) at the destination before they can be delivered to the upper layers.

For comparison, also shown in Fig. 1(b) is the corresponding delay performance when sender-side coding is used along with maximum throughput routing. The dramatic improvement in delay performance is evident.

II. Routing and Coding

A. *Low-delay fountain coding*

Maximum throughput routing maximises the rate at which packets are delivered to a destination, but provides no guarantee of in-order arrival at the destination (quite the opposite

Manuscript received November 26, 2008. The associate editor coordinating the review of this letter and approving it for publication was H. Yousefizadeh.

This work is supported by Science Foundation Ireland grant IN3/03/I346.

The authors are with the Hamilton Institute, National University of Ireland, Maynooth, Co. Kildare, Ireland (e-mail: {badarla.venkataramana, vijay.subramanian, doug.leith}@nuim.ie).

Digital Object Identifier 10.1109/LCOMM.2009.081997

Algorithm 1 Max-throughput routing at node n, time t

```
1: For each neighbour $m$, compute the utility $U_{n,m}(t) = \sum_{m \in N_n} (q_n(t) - q_m(t)) \times R_{n,m}$. $R_{n,m}$ is the link rate from $n$ to its neighbour $m$, $q_n$ the number of packets queued at node $n$, $N_n$ the set of neighbours of node $n$.
2: Determine $m$ that maximises $U_{n,m}(t)$. Denote by $m^*$.
3: Transmit min$(q_n(t), R_{n,m^*})$ packets to node $m^*$.
```

in fact, as the previous example illustrates). An ideal fountain code enables n information packets to be reconstructed from *any* $n + \delta$ coded packets, with overhead δ small. One immediate consequence is that a fountain coded packet stream is insensitive to packet reordering. The use of fountain codes in combination with maximum throughput routing therefore offers the potential for significant gains in delay performance while yielding excellent throughput performance. It is this key observation that motivates the approach studied in this paper.

We note, however, that current practical (as opposed to ideal) fountain codes, for example LT codes [5] and Raptor codes [6], are not suitable for our purpose. One feature of these practical codes is that large block sizes n are required in order to obtain reasonably small overhead δ. For example, block sizes of 10K packets and larger are commonly considered in the literature. For smaller block sizes, the overhead quickly becomes large e.g. even highly optimised LT codes can have overheads of greater than 40% [7]. The reason that block size matters is that all of the information packets are generally only recovered once a complete block (i.e. $n + \delta$) of coded packets has been received. That is, the coding introduces a decoding delay that is proportional to the block size. For large block sizes, the decoding delay is large and so the net delay performance of joint coding/routing may not be any better than that with routing alone. We therefore begin by considering the design of low-delay, small block-size fountain codes.

In LT codes a coded packet e_i is generated from the sum of a randomly selected subset of the information packets. That is, $e_i = g_i u$ where u is the vector of information packets and g_i is a vector with 0 or 1 entries – in modulo 2 arithmetic summing two packets simply corresponds to xoring the corresponding bits in each packet and so is computationally cheap. Raptor codes are similar, except that they make use of intermediate coded symbols. Stacking $n + \delta$ received coded packets into a vector e, we have that $e = Gu$ where row i of G is the (0,1) vector g_i. The information packets can be recovered (i.e. decoded) whenever sufficient coded packets are received that the matrix G is full rank. Decoding involves solving n linear equations $e = Gu$ which in general is an $O(n^3)$ operation using, for example, Gaussian Elimination (GE).

A key design driver for LT and Raptor codes is the requirement for efficient decoding e.g. linear time $O(n)$ decoding. This is vital when large block sizes n are used and leads to the use of matrices G which are sparse so as to be decodable using Belief Propagation (BP). However, our interest is in small block sizes n in order to ensure small decoding delay. When n is small, we can afford more expensive decoding algorithms such as GE decoding. This allows us to make use of matrices G which are non-sparse.

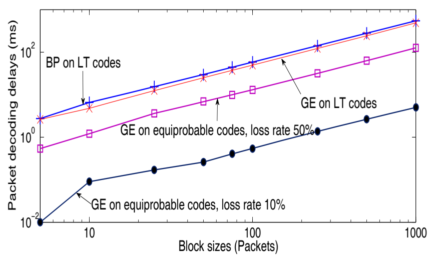


Fig. 2. Decoding delay of LT (using both BP and GE decoding) and equiprobable codes (using GE decoding).

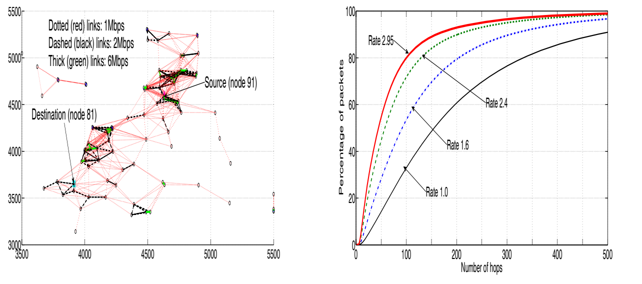
In particular, we consider generating coded packets as follows. First the n information packets are transmitted unencoded (i.e. we use a systematic code). Subsequent coded packets are then formed by tossing a coin for each information packet and xoring the selected packets. That is, each element in vector g_i takes value 1 with probability 0.5 (and so also takes value 0 with probability 0.5, hence 0 and 1 are equiprobable). It is shown in [8] that this class of fountain codes has lower overhead δ (and so decoding delay) than any sparse code. Use of a systematic code further reduces decoding delay when the level of packet re-ordering is low. Fig. 2 illustrates the superior decoding delay of this code in comparison with LT codes using BP decoding and LT codes using GE decoding.

B. *Pipelining blocks*

When using small blocks, transmission of a large file requires the use of multiple blocks. We take advantage of pipelining of blocks to reduce the coding overhead further. Specifically, once the initial uncoded packets of a block are transmitted, we wait for an *ack* from the destination before sending coded packets. While waiting for the *ack* we commence sending packets from the next block, if available. The destination sends an *ack* for each received packet. The *ack* indicates the block j to which the packet belongs, the block k next in line for decoding, the rank of the currently received G matrix for block k. On receiving an *ack* with j not equal to k, indicating reordering/loss crossing block boundaries, the sender transmits coded packets from block k. The number of coded packets sent is determined by the rank of the already received G matrix. Once these have been sent, say at time t, to allow for the network round-trip time no further coded packets are transmitted until a packet sent after time t is *ack*ed.

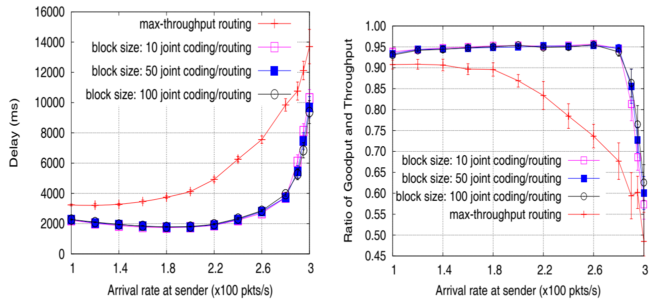
C. *Routing*

When block k is finally decoded at the destination, due to pipelining of transmissions there may still be packets associated with that block in-flight within the network. Since forwarding of these packets uses network resources without yielding benefit it makes sense for forwarders to simply drop these packets once an *ack* indicates that a block is decoded. However, this violates a key assumption in the stability results for maximum throughput routing: namely, that the set of available routing actions at time t is independent of previous



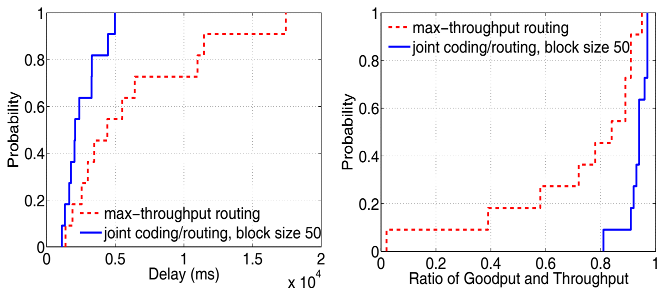
(a) Roofnet topology (b) Effects of routing loops on maximum throughput routing

Fig. 3. Roofnet-like topology (91 nodes, node locations derived from Roofnet GPS data, two-ray ground path propagation model).



(a) Delay (b) Goodput

Fig. 4. Roofnet topology, flow between nodes 81 and 91.



(a) Delay (b) Goodput Efficiency

Fig. 5. Roofnet topology, average over 12 randomly selected source-destination pairs. Poisson arrival rate 90% of capacity.

routing actions. To preserve stability guarantees we make routing decisions based on virtual queues that mirror the real queues except that in-flight packets are not dropped from the virtual queues. Standard stability results then ensure that the virtual queues within the network are guaranteed to be bounded. Since the virtual queues sizes always upper bounds the real queues sizes, it follows that the real queues are also guaranteed to be bounded i.e. network stability is ensured.

III. Performance

For the simple network topology in Fig. 1(a), the improvement in delay performance when combined coding/routing is used can be seen in Fig. 1(b). We also consider the more realistic topology shown in Fig. 3.(a), which is related to the Roofnet network. To demonstrate the effect of routing loops, we measured the number of hops traveled by each packet of a flow running from node 91 to node 81 and plot the hop count distribution in the Fig. 3.(b). We can observe that, at high arrival rates 50% of packets travel more than 50 hops, while at low rates, 50% of packets travel more than 150 hops. As the packets are traveling significantly higher hops when compared to the network diameter (10 hops) and the shortest distance between the nodes 91 and 81 (6 hops), we can infer that routing loops indeed lead to considerable packet delays.

Figs. 4.(a) and (b) plot the corresponding mean packet delay and ratio of goodput and throughput, respectively for a range of offered loads and block sizes. It can be seen that joint routing/coding consistently achieves significantly improved delay performance compared with plain maximum throughput routing; this also shows that performance is insensitive to the choice of block size over a wide range of arrival rates. While goodput takes account of the in-order packet delivery to upper layers at the destination, throughput provides a measure of maximum achievable network capacity for a given flow configuration. We can observe that, for joint coding/routing, in most cases (except at high loads 2.95 and 3.0, the maximum network capacity), goodput is about 95% of throughput (i.e., 5% of overhead for in-order delivery), which confirms that the improved delay performance with coding is not achieved at the cost of high coding overhead and reduced network capacity.

Fig. 5 shows the delay and goodput distributions taken over 12 different source-destination pairs with the arrival rate fixed at 90% of network capacity between each pair. Once again, a substantial improvement in delay performance is consistently observed with joint routing/coding.

IV. Conclusions

In this paper we highlight some fundamental difficulties with current maximum throughput routing algorithms, in particular a tendency towards extensive routing loops and poor delay performance. We propose a joint routing/coding approach and demonstrate that this achieves significantly improved delay performance while maintaining excellent throughput performance.

References

[1] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling for maximum throughput in multihop radio networks," *IEEE Trans. Automatic Control*, vol. 37, no. 12, pp. 1936-1949, 1992.

[2] A. L. Stolyar, "Maximizing queueing network utility subject to stability: greedy-primal dual algorithm," *Queueing Systems*, vol. 50, no.4, pp. 401-457, 2005.

[3] M. J. Neely *et al.*, "Fairness and optimal stochastic control for heterogeneous networks," in *Proc. IEEE Infocom*, 2005.

[4] B. Radunovic *et al.*, "An optimization framework for opportunistic multipath routing in wireless mesh networks," in *Proc. IEEE Infocom mini symposium*, pp. 2252-2260, 2008.

[5] M. Luby, "LT codes," in *Proc. 43rd Annual IEEE Symposium on Foundations of Computer Science*, pp. 271-280, 2002.

[6] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2551-2567, 2006.

[7] E. Hyytia *et al.*, "Optimal degree distribution for LT codes with small message length," in *Proc. IEEE Infocom*, 2007.

[8] V. G. Subramanian and D. J. Leith, "On a class of optimal rateless codes," in *Proc. Allerton Conference*, 2008.