AUTONOMOUS ROUTING ALGORITHMS FOR NETWORKS WITH WIDE-SPREAD FAILURES

Wajahat Khan, Long Bao Le and Eytan Modiano

Communications and Networking Research Group
Massachusetts Institute of Technology
Cambridge, MA, USA 02139
Emails: {wajahat, longble, modiano}@mit.edu

***Abstract*— We study end-to-end delay performance of different routing algorithms in networks with random failures. Specifically, we compare delay performances of Differential Backlog (DB) and Shortest Path (SP) routing algorithms and show that DB routing outperforms SP routing in terms of throughput when the network is heavily loaded and/or the failure rate is high while SP routing achieves better delay performance in the low load regime. Then, we investigate delay performance of a hybrid routing algorithm that combines principles of both SP and DB routing algorithms and show that it outperforms both of these routing algorithms. Finally, we demonstrate improvements in delay performance of DB routing through the use of a digital fountain approach which was originally proposed for multicast applications. In addition, our results show that there exists an optimal coding rate where digital fountain based DB routing achieves minimum end-to-end delay. To the best of our knowledge, this is the first work which investigates delay performance of DB routing and its enhanced versions for networks with link failures.**

***Index Terms*—Differential backlog routing, shortest path routing, digital fountain, end-to-end delay, throughput region**

I. Introduction

A robust communications network must have built-in redundancies to recover from failures. However, survivable topology design forms a necessary but not sufficient condition for failure recovery. Recovering from a failure usually involves re-routing of data along pre-planned back-up paths (typically used for recovery from single link failures) [1], [2], [3]. However, in the event of a catastrophic failure, the use of pre-planned back-up paths is not practical. When dealing with networks on the scale of a nation-wide communications infrastructure, it is impossible to pre-plan against all the possible link failures. For example, the number of possible failure scenarios for a graph with just thirty links is over a billion. Also, for such an approach to work, every node in the network must know the state of the entire network which may not be achieved in most practical networks. Hence, autonomous re-routing algorithms which can dynamically respond to failure(s) and require minimal network-state knowledge are a key solution for networks with failures.

Traditional approaches to autonomous routing have been based on shortest path algorithms such as Open Shortest Path First (OSPF) [4]. These algorithms have their drawbacks when responding to changes in network topologies (which includes failures) both in terms of efficient capacity utilization and speed of recovery. The time complexity of computing all-pair shortest paths using a simple algorithm like Bellman-Ford algorithm is $O(N^3)$ where N is the number of nodes [5]. For large networks this computation over short time periods becomes not only challenging but practically impossible. Distributed implementations which are necessary in a nation-wide network have yet greater time complexity. Our goal in this paper is to develop novel approaches to the autonomous re-routing problem for rapid and efficient failure recovery.

The DB routing algorithm was originally proposed by Tassiulas and Ephremides in the context of wireless networks which was shown to achieve the maximum throughput performance [6]. DB routing can be easily implemented in a distributed manner in wired networks, such as optical backbone networks, since each node only needs to know the states of its neighbors. The throughput-optimal performance and ease of implementation of DB routing makes it an obvious candidate for autonomous re-routing. Although recent testbed implementation shows that DB routing achieves good throughput and fairness performance, its delay performance is not well understood [11].

In this paper, we investigate delay performance of DB

This work was supported by NSERC Postdoctoral Fellowship and by ARO Muri grant number W911NF-08-1-0238, DTRA grant number HDTRA1-07-1-0004, NSF grant numbers CNS-0626781 and CNS-0830961.

routing and its enhanced variants in an optical backbone network of major US service providers. Specifically, we compare end-to-end delay performance of DB and SP routing in this backbone network with and without failures. We show that SP routing indeed achieves lower delay in the low load condition but it has much lower throughput performance compared to the DB routing. We then compare delay performance of a hybrid (HybridDB) routing algorithm which combines the principles of DB and SP routing algorithms by incorporating routing costs into the "differential backlog" routing metric. We show that the HybridDB routing algorithm indeed outperforms both DB and SP routing algorithms while achieving the maximum throughput performance. Given the fact that long end-to-end delay of a few packets from a file results in significant increase of file transfer delay, we propose to combine the digital fountain approach with DB routing to reduce to the file transfer delay. Our results show that digital fountain based DB routing successfully improves end-to-end file transfer delay and there exists an optimal coding rate that achieves the minimum delay performance.

This paper is organized as follows. In section II, we briefly review the SP and DB routing algorithms. We also describe the HybridDB routing and the digital fountain based DB routing algorithms. In section III, we describe our simulation setup and present extensive simulation results which compare performance of the aforementioned routing algorithms. Conclusions are stated in section IV.

II. System Model and Routing Algorithms

In this section, we describe the system model, review the SP and DB routing algorithms and present two enhanced routing algorithms, namely HybridDB and digital fountain based DB routing algorithms whose performances are investigated in the next section.

A. System Models

We consider a wired network (e.g., optical backbone networks) which is formed by a set of nodes and links. Assume that the network has N nodes. It is further assumed that time is slotted. Assume that files arrive randomly over time which are destined to randomly chosen destination nodes. For any such arriving file, a routing algorithm is employed for end-to-end data delivery. In the following, we refer to a combination of such a file, its source and destination nodes as a data session. We will consider four different routing algorithms, namely SP, DB, HybridDB and digital fountain based DB routing algorithms. Among these routing algorithms, SP is a single-path routing policy while the others are multipath routing policies.

Data packets of different sessions are buffered at each network node according to their destinations. Specifically, each node maintains $N-1$ FIFO queues to buffer data packets which are destined to other $N-1$ nodes. A destination node of any file/session waits until it receives a sufficient number of packets to reconstruct the underlying file. After the file is reconstructed, it is delivered to the higher layers and exits the network.

We are interested in end-to-end packet and file transfer delay performances of the aforementioned routing algorithms where end-to-end packet/file delay is measured from the time instant a packet/file enters the network to the time instant it is successfully received/reconstructed at the destination node. Note that a file exits the network only after the destination node receives a sufficient number of packets to reconstruct it. It is assumed that network links are either in "WORKING" or "FAILED" states in any time slot. A "WORKING" link can transmit one packet/time slot while a "FAILED" link cannot transmit any packet until its state is changed to the "WORKING" state. We assume that network failures only impact network links. Because data packets are buffered at node queues, there is no need for error recovery at the link or transport layer.

B. Routing Algorithms

1) SP Routing Algorithm: SP routing is by far the most common class of algorithms employed in modern networks. As might be implied from its name, SP routing considers all the possible paths between a source and destination and chooses the one with minimum cost. The costs are typically a function of length, congestion or monetary costs of a link. Bellman-Ford and Dijkstra [5] are two widely used shortest path algorithms. The best-known running times of these algorithms are $O(NE)$ and $O(N \log N + E)$ respectively, where N represents the number of nodes and E represents the number of links in a network. In this paper, we assume that routing costs of the SP routing algorithm are simply the number of hops along a route (SP routing algorithm is simply minimum-hop routing).

2) DB Routing Algorithm: The underlying idea behind DB routing is to use all network resources to distribute data, differentiated by destination (data destined to one particular destination is called the corresponding commodity). Routing decisions are made at the beginning of each time slot. To describe the DB routing in more details, let $U_a^{(c)}(t)$ be the number of packets waiting at node a destined for node c in time slot t. For each pair of directly connected nodes, let us say nodes a and b, the commodity $c_{ab}^*(t)$ with the highest differential backlog which is calculated as

$$c_{ab}^*(t) = \underset{c \in \{1,...,N\}}{\operatorname{argmax}} \left\{ U_a^{(c)}(t) - U_b^{(c)}(t) \right\} \quad (1)$$

is transmitted over the link (a, b) in time slot t where $\{1, ..., N\}$ is the set of network nodes. In the case where there are multiple commodities achieving the maximum differential backlog, one of them is chosen randomly. Also, if the number of packets at a particular queue for some commodity is smaller than that required by the routing algorithm, outgoing links of the corresponding node are randomly chosen to transmit available packets.

It has been shown that the DB routing algorithm achieves maximum throughput performance [6], [7]. Specifically, let throughput region be the union of all flow/sessions arrival rate vectors such that there exists some routing algorithm which can stabilize all the network queues (i.e., their queue lengths are bounded). It can be shown that DB routing algorithm stabilizes all arrival rate vectors which lie strictly inside the throughput region [6], [7]. Although DB routing is throughput optimal, its delay performance is not well-understood. In fact, some recent works show that the maximum weight scheduling (a version of DB routing for single-hop wireless traffic flows) achieves order optimal delay in a wireless cellular network and most practical wireless ad hoc networks [9], [10]. However, its actual delay performance was not investigated in these papers.

3) HybridDB Routing Algorithm: It has been recognized that although DB routing algorithm is throughput-optimal, its delay performance may not be very good in low network load conditions [7], [11]. This is because the DB routing algorithm exploits all possible routes including loops in the network to achieve the maximum throughput. In the low load regime, the DB routing algorithm tends to use long routes [11] which may hurt the delay performance. In fact, routing data along short routes may achieve good delay performance while still maintaining queue stability (i.e., bounded queue length) in the low network load. Therefore, an adaptive DB routing which uses long routes only when necessary can potentially achieves both good throughput and delay performances.

One way to exploit advantages of both DB and SP routing algorithms is to incorporate routing cost into the differential backlog metric presented in (1) so that data is routed along short routes when the network load is low. Specifically, we propose the HybridDB routing algorithm which chooses a commodity for link (a, b) in time slot t as follows [7]:

$$c_{ab}^*(t) = \underset{c \in \{1,...,N\}}{\operatorname{argmax}} \Big\{ U_a^{(c)}(t) - U_b^{(c)}(t) + \alpha \left(V_a^{(c)}(t) - V_b^{(c)}(t) \right) \Big\} \quad (2)$$

where $V_i^{(c)}$ is the routing cost to deliver data from node i to node c and α is a weighting factor. If the underlying SP routing is minimum-hop routing then $V_i^{(c)}$ is simply the number of hops on the minimum-hop route from node i to node c and $V_a^{(c)}(t) - V_b^{(c)}(t) = 1$. In addition, the larger the weighting factor α the more likely short routes are selected for data delivery.

4) Digital Fountain Based DB Routing Algorithm: For end-to-end delivery from a source node to the corresponding destination node, a long file must be broken into a large number of packets. Because DB routing may use a long route to deliver a packet, it is highly likely that it takes a long time before the destination node receives all required packets to reconstruct the file. Hence, the end-to-end file transfer delay could be potentially reduced if the destination node can reconstruct the file by using only a subset of the original packets. In fact, the digital fountain coding technique enables us to achieve this goal. The digital fountain approach was originally proposed by Byers, Luby and Mitzenmacher for the asynchronous multicast application [12].

In an ideal digital fountain approach, x packets broken from a file are encoded into $n > x$ packets which are then transmitted over the network. A receiver which receives any x distinct packets out of the n transmitted packets in any order can reconstruct the original file. This ideal digital fountain approach can be realized by using the classical Reed Solomon (RS) erasure code. However, it has been indicated in [12] that the ideal digital fountain using RS erasure code has several implementation limitations. The authors of [12] have also proposed several codes including Tornado codes [13] and Luby Transform codes [14] which can approximate the ideal digital fountain and are easy to implement. The approximate digital fountain solution usually requires the number of received packets to be slightly larger than the number of original packets before a receiver can reconstruct the original file.

To make our investigation independent of code design issues, we assume an ideal digital fountain solution in this paper. It can be observed that digital fountain approach can be jointly employed with any routing algorithms presented in the previous subsections. This is because the digital fountain approach is only needed to encode data at a source node and to reconstruct original files at a destination node.

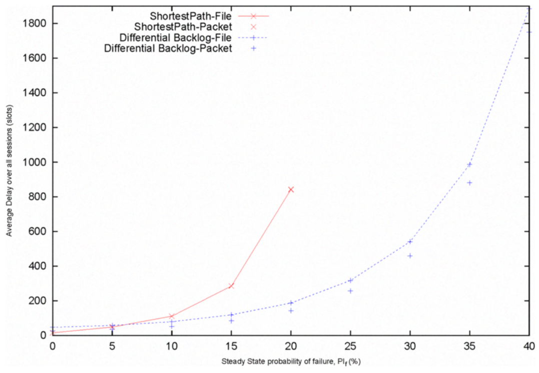

Fig. 1. File and packet delays of SP and DB versus failure rate (for λ=1/4, p= 1/4, simulation time: 1 million slots)

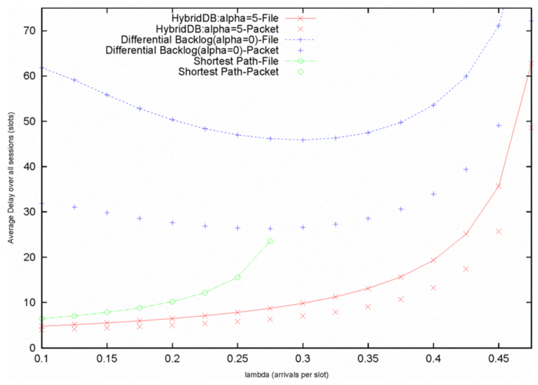

Fig. 2. Delay performance of SP, DB, and HybridDB versus network load (for p= 1/4, simulation time: 10 million slots)

III. Performance Evaluation

In this section, we investigate and compare delay performances of the routing algorithms presented in the previous section. All numerical results are obtained for an optical backbone network of major US service providers with 29 bidirectional links and 13 nodes [8].

A. Simulation Settings and Parameters

Recall that time is slotted. Files arrive at the beginning of each slot at each node according to a Poisson process with arrival rate λ files per slot. A file is equally likely to be destined to any node in the network except the source node. The number of packets in each file is a geometric random variable with parameter p. In order to investigate the impacts of code rate on delay performance of different routing algorithms using the digital fountain solution, we fix file sizes at x packets (in Figs. 6, 7). Also, for a file of x packets, $n = \lceil x/f \rceil$ packets are generated where f is the code rate. At the destination node, a file reconstruction is assumed to be performed successfully when any x of the $n = \lceil x/f \rceil$ packets originally transmitted by source node have been received.

Nodes keep track of the number of packets they have received for each file destined to them. Since there is no packet loss, redundant packets in a file do get to the nodes sometimes whereby they are discarded. A file is called active if its destination has not received all the packets required to reconstruct that file. A list of all active files, is maintained at each destination node.

For DB-based routing algorithms, after all arrivals for a slot take place, each link is marked with the commodity that has the maximum differential backlog across that link, with ties broken randomly. For links with a positive differential backlog, one packet of the marked commodity is transmitted across the link. If there

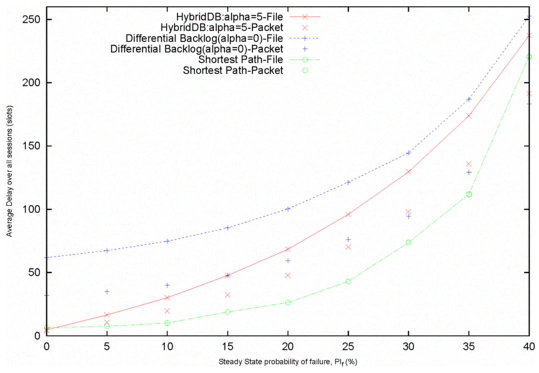

Fig. 3. File and packet delays of SP, DB, and HybridDB versus failure rates (for λ=1/10, p= 1/4, simulation time: 10 million slots)

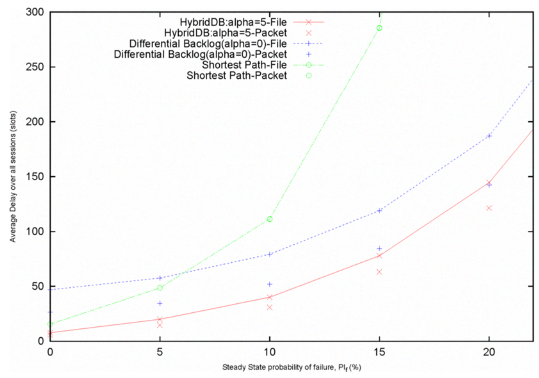

Fig. 4. File and packet delays of SP, DB, and HybridDB versus failure rates (for λ=1/4, p= 1/4, simulation time: 10 million slots)

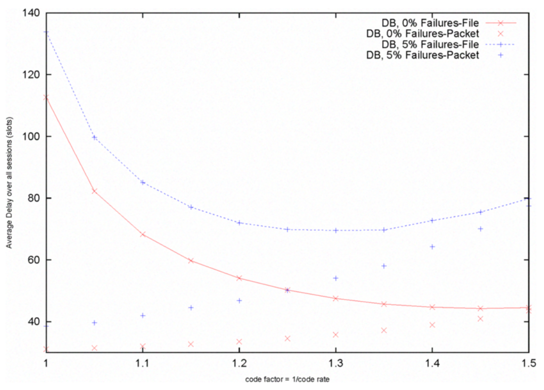

Fig. 5. Delay performance of Digital Fountain approach in DB versus code rate (for λ=1/25, x=20, simulation time: 1 million slots)

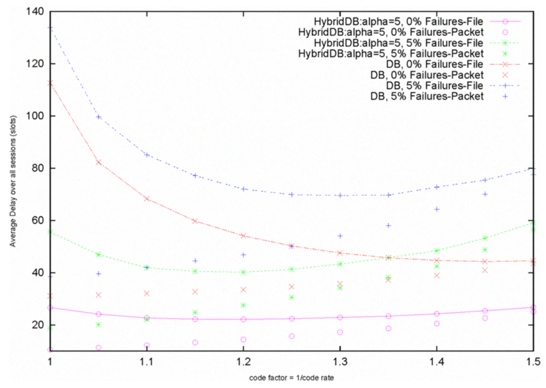

Fig. 6. Delay performance of Digital Fountain approach in DB and HybridDB versus code rate with and without failures (for λ=1/25, x=20, simulation time: 1 million slots)

is a shortage of commodity packets at a network node, the competing links are randomly served. All packet transmissions are completed by the end of a slot.

At the beginning of each slot, a "WORKING" link changes its state to "FAILED" with probability p_f and a "FAILED" state changes it state to "WORKING" with probability p_w (i.e., link status is modeled as a two state Markov chain). All links start in the "WORKING" state. Hence, the steady state probability, π_f , of being in the "FAILED" state is given as

$$\pi_f = \frac{p_f}{p_f + p_w}. \tag{3}$$

For SP routing, we use Dijkstras All-pair Shortest-Path algorithm. As each link can transfer up to one packet in each slot, the cost of all links are set to be equal to 1. Each node maintains an address classifier to route packets to outgoing links based on their destinations. Queues at network nodes buffer new arrivals to the nodes as well as routed packets from other parts of the network. The address classifiers are updated every time slot to account for any changes in network topology resulting from any link failures or activations. When a link fails, all packets residing in its queue get rerouted to the link source. End-to-end delay are averaged over all sessions.

B. Comparison of SP, DB, and HybridDB Routing

In Fig. 1, we show delay performances of SP and DB routing algorithms versus the steady state probability of "FAILED" state (called failure probability in the following). It can be observed that in the SP routing, packets are routed deterministically toward their destinations; hence SP routing behaves better in terms of delays when the network load is low (i.e., low failure probability). However, since SP routing does not utilize all possible routing paths in the network to perform load balancing,

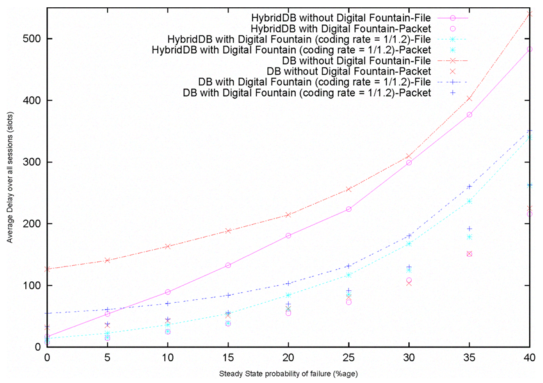

Fig. 7. Delays of Digital Fountain approach in SP, DB and HybridDB versus failure rate with different values of code rate (for λ=1/50, x = 20, simulation time: 1 million slots)

it does not achieve the full throughput of the network. Therefore, DB routing achieves much greater throughput than SP routing. This can be confirmed by noticing that file transfer delay under SP routing increases sharply when failure probability is larger than 15% while rapid increase in file transfer delay under DB routing only occurs for failure probability larger than 35%.

In Figs. 2, 3, 4, we compare delay performances of SP, DB, and HybridDB routing algorithms. These figures show that HybridDB routing achieves better delay performance than DB routing. It can also be seen from Fig. 2 that DB and HybridDB routing algorithms have similar throughput performance. This can be interpreted as follows. HybridDB routing can exploit all possible paths to achieve maximum throughput when the network load is high but it tends to use shorter and more direct paths to deliver data in the low load condition. Fig. 3 shows that for networks with link failures, the delay improvement of HybridDB routing compared to DB routing becomes less significant when the failure rate is high. In addition, SP routing has good delay performance in the low load condition but it achieves lower throughput compared to the other two routing algorithms.

C. Routing Performance with Digital Fountain Solution

In Fig. 5, we show delay performance of DB routing with the digital fountain solution versus code factor which is equal to 1/code rate for networks with and without failures. It can be observed from this figure that end-to-end file transfer delay of DB routing decreases when the code factor increases from 1; then it increases when the code factor reaches a certain optimal value. In addition, end-to-end packet delay always increases when the code factor increases for both networks with and without failures. These results can be interpreted as follows. With the increase in the code factor, source nodes add increasing amount of redundant information into the original data. Addition of more redundant information would reduce the end-to-end file transfer delay for low code factors because a receiver can reconstruct an original file by using a smaller fraction of the encoded file. However, adding more redundant information also increases network load and, therefore, congestion in the network which would potentially increase queueing delay. Therefore, there exists an optimal code factor which depends on network load and failure rates. In contrast, packet delay always increases with increased code factor because individual packets do not benefit from the use of coding.

In Fig. 6, we compare delay performances of DB and HybridDB routing algorithms with the digital fountain solution for different values of code factors. This figure shows that by employing the digital fountain solution, HybridDB routing can further improve delay performance compared to the original DB routing. Also, there exists an optimal code factor for the HybridDB routing which is smaller than the optimal code factor under the DB routing. This is because the HybridDB routing algorithm tends to use shorter routes which makes the network "more congested" compared to the DB routing algorithm. Finally, we compare delay performances of DB and HybridDB routing algorithms with and without the digital fountain solution versus failure rates in Fig. 7. Again, HybridDB routing using the digital fountain solution achieves the best file transfer delay performance compared to other algorithms.

IV. Conclusions

In this paper, we investigated delay performances of the DB routing algorithm and its enhanced versions for networks with link failures. Specifically, we have shown through extensive numerical investigation that SP routing achieves good delay performance in the low network load regime but it has very low throughput performance compared to the DB routing algorithm. In addition, by combining the principles of both SP and DB routing algorithms, HybridDB routing has better delay performance than DB routing while achieving similar throughput performance. Moreover, the digital fountain approach can be combined with DB or HybridDB routing to further improve end-to-end file transfer delay. Finally, there exists an optimal code factor which results in the minimum end-to-end file transfer delay which depends on the network load and failure probability.

References

[1] R. Bhandari, Survivable Networks: Algorithms for Diverse Routing, Kluwer Academic Publishers, 1999.

[2] E. Modiano and A. Narula-Tam, "Survivable lightpath routing: A new approach to the design of WDM-based networks," *IEEE J. Sel. Areas Commun.,* vol. 20, no. 4, pp. 800–809, 2002.

[3] A. Narula-Tam, E. Modiano, and A. Brzezinski, "Physical topology design for survivable routing of logical rings in WDM-based networks," *IEEE J. Sel. Areas Commun.,* vol. 22, no. 8, pp. 1525–1538, 2004.

[4] Christian Huitema, Routing in the Internet, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, Second Edition. The MIT Press, September 2001.

[6] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Aut. Control,* vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

[7] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time varying wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 1, pp. 89–103, Jan. 2005.

[8] W. F. Khan, "Autonomous routing algorithms for networks with wide-spread failures: A case for differential backlog routing." Master thesis, Massachusetts Institute of Technology, 2008.

[9] M. Neely, "Delay analysis for max weight opportunistic scheduling in wireless systems," *Allerton 2008*, Sept. 2008.

[10] L. B. Le, K. Jagannathan, and E. Modiano, "Delay analysis of maximum weight scheduling in wireless ad hoc networks," *CISS'2009*, Mar. 2009.

[11] A. Warrier, S. Janakiraman, and I. Rhee, "DiffQ: Practical differential backlog congestion control for wireless networks," *INFOCOM 2009.*

[12] J. W. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE J. Sel. Areas Commun.,* vol. 20, no. 8, pp. 1528–1540, 2002.

[13] M. Luby, "Information additive code generator and decoder for communications systems," U.S. Pat. No. 307 487, Oct. 2001.

[14] M. Luby, M. Mitzenmacher, A. Shokrollahi, and D. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Inf. Theory,* vol. 47, pp. 569–584, Feb. 2001.