51st IEEE Conference on Decision and Control
December 10-13, 2012. Maui, Hawaii, USA

Social Networks over Wireless Networks

Eleni Stai[1], John S. Baras[2], Symeon Papavassiliou[1]

Abstract— **We consider the formation, operation and maintenance of dynamic social networks (among human users) supported by technological communication networks such as wireless networks, or hybrid wireline-wireless networks. The technological (physical) networks of interest display dynamic behavior in several dimensions, including variable connectivity, variable congestion, variable link characteristics. As broadband wireless devices and networks are becoming ubiquitous these human-machine systems, that combine the social aspects and behavioral activities of humans with the technological characteristics of the underlying physical networks, provide several important challenges in efforts to model them, evaluate their performance and dynamically control them so certain performance requirements are met. These include combinations of performance, trust, privacy, energy efficiency. In this paper we develop novel models for these complex human-machine systems that incorporate social network behavioral models and wireless network models that are inspired from statistical physics (hyperbolic graphs). We investigate the performance of wireless network protocols that support and respond to the constraints implied by the social network they support.**

I. Introduction and Related Work

Nowadays, an increasing number of people own their personal smart phones, laptops or tablets and use them in ad hoc mode, forming point-point communications supporting different applications such as file exchange, online games, GPS connections, etc [1], [2]. Moreover companies develop social network applications for ad hoc networks [1], in order to be used in large conferences, university libraries, university campuses, etc. These social network applications involve real time applications such as chatting and video. However, real time applications over wireless multihop networks, demand routing/scheduling algorithms that achieve desirable delay-throughput trade-offs, with high throughput and low end-to-end delay. In this paper we focus on the performance analysis (throughput, delay) of wireless network protocols that support and respond to heterogeneous social networks.

This interaction between social and wireless networks has driven a recent research interest on the design of efficient wireless topologies and algorithms that take into consideration the social character of the applications they are about to support. In [3], a utility network maximization algorithm for resource allocation for social wireless networks is proposed, where the utility function is enhanced in order to include the social distance between the source-destination pairs. In [4] a topology modification framework, for the design of social-friendly wireless networks, is developed. In a similar spirit, the authors in [5] develop an algorithm for file sharing for social wireless networks that takes into account the acquaintances of the requesting nodes on the overlay social network and improves the scaling performance of the file dissemination. Last but not least, the authors in [6] study the capacity of composite wireless and social networks.

As aforementioned, in this work, we are focusing on the algorithms for scheduling and routing in a wireless network with social traffic. The backpressure algorithm, introduced in its original form in [7], has received much attention by the research community in the past few years (i.e. [8], [9]), as it satisfies the throughput optimal requirement. The backpressure algorithm performs routing and scheduling based on congestion gradients, by allowing transmission to the links that maximize the differential backlog between neighboring nodes (max-weight matching). However, by deploying routing without using any information about the position or distance to the destination, it explores all possible source-destination paths leading to undesirable high delays even in the case of light traffic. Several approaches have been developed for reducing the delay imposed by the pure backpressure scheduling/routing [9], [10], [11].

In this paper, we propose a social and delay aware backpressure scheduling/routing algorithm which admits as input a network embedded in hyperbolic space via the greedy embedding of [12]. A greedy embedding in hyperbolic space is a correspondence between nodes and hyperbolic coordinates such that the greedy routing algorithm, employed in hyperbolic coordinates, does not have local minima, i.e. every node can find at least one neighbor closer than itself to all possible destinations [12], [13]. In [12], a distributed implementation of a greedy embedding is proposed, which can assign hyperbolic coordinates to new nodes without re-embedding the whole network. In this work, we impose routing constraints on the backpressure algorithm, by determining as next-hop neighbors for a specific destination only "greedy" neighbors, i.e. those that strictly reduce the hyperbolic distance to the destination. Simultaneously, we consider an overlay social weighted graph which is also involved in the scheduling and routing procedure through its socially determined weights. We propose two algorithms, the "Greedy" backpressure and the "Greediest" backpressure for both static and dynamic networks. The first one performs routing by choosing as next hop node one of the greedy neighbors, while the Greediest backpressure chooses the

[1] Eleni Stai (estai@netmode.ntua.gr) and Symeon Papavassiliou (papavass@mail.ntua.gr) are with the School of Electrical and Computer Engineering, National Technical University of Athens (NTUA), Athens, Zografou, 15780, Greece. This research was performed while Eleni Stai was pursuing an Internship at the University of Maryland, College Park.

[2] John S. Baras (baras@umd.edu) is with the Institute for Systems Research and the Department of Electrical & Computer Engineering, University of Maryland, College Park, MD 20742, USA.

978-1-4673-2063-4/12/$31.00 ©2012 IEEE

greedy neighbor with the least hyperbolic distance to the destination. Both algorithms perform backpressure scheduling and as a result routing/scheduling is based on both congestion and distance gradients.

The rest of the paper is organized as follows. Section II provides some insights on greedy embeddings and the hyperbolic space, while section III describes the system model. Sections IV, V, VI, VII, describe, analyze and extend the proposed algorithms. Finally, in Section VIII the efficiency of our algorithms is verified through simulations and Section IX concludes the paper.

II. Greedy Network Embedding in Hyperbolic Space

Greedy embeddings in other than Euclidean metric spaces, have been proposed to improve greedy routing techniques. In the simplest form of greedy routing, the sender forwards the message to its directly (one-hop) connected neighbor which reduces the most the distance to the destination. In [12], a distributed implementation of a greedy embedding in hyperbolic space is proposed, which can be also applied in dynamic network conditions, by assigning hyperbolic coordinates to new nodes without re-embedding the whole network. Before describing the embedding algorithm of [12], we briefly introduce some concepts of the two-dimensional hyperbolic space that will be used in this paper.

The whole infinite hyperbolic plane can be represented inside the finite unit disc $\mathbb{D} = \{z \in \mathbb{C} || z| < 1\}$ of the Euclidean space; the Poincaré Disc model. The greedy embedding used in this work is based on the Poincaré Disc model. The hyperbolic distance function $d_H(z_i, z_j)$, for two points z_i, z_j, in the Poincaré Disc model is given by [12], [14]:

$$\cosh d_H(z_i, z_j) = \frac{2|z_i - z_j|^2}{(1-|z_i|^2)(1-|z_j|^2)} + 1 \tag{1}$$

The Euclidean circle $\partial\mathbb{D} = \{z \in \mathbb{C} || z| = 1\}$ is the boundary at infinity for the Poincaré Disc model. In addition, in this model, the shortest hyperbolic path between two nodes is either a part of a diameter of $\mathbb{D}$, or a part of a Euclidean circle in $\mathbb{D}$ perpendicular to $\partial\mathbb{D}$.

The greedy embedding is constructed by choosing a spanning tree of the graph of the initial network and then embedding the spanning tree into the hyperbolic space according to the algorithm of [12]. Following this algorithm, after having assigned hyperbolic coordinates to the root of the tree inside a specific hyperbolic area, each node computes its own coordinates using the ones of its parent, in such a way that the hyperbolic bisector of the embedded spanning tree edge between the node and its parent, does not intersect any other embedded edge of the spanning tree. This property, which holds only for the embedded edges of the spanning tree (not for the non-spanning tree edges), is sufficient for the embedding of the network in the hyperbolic space to be greedy. As the computation of the virtual coordinates of a node depends only on its parent's coordinates, this algorithm for greedy embedding can be deployed distributively by a wireless ad hoc network which lacks any centralized infrastructure. If a spanning tree of the graph admits a greedy embedding in hyperbolic space then the whole graph can also be embedded [13]. Let us denote as "greedy" paths, the paths consisting of nodes with strictly diminishing distances to the destination. From definition, the greedy embedding ensures the existence of at least one greedy path between each source-destination pair in the case of static networks. Every pair of nodes i, j is connected through a unique path, let us denote it as $i,\ i_1,\ i_2, \ldots\ i_k,\ j$, lying on the spanning tree which is embedded in the hyperbolic space. Due to the particular embedding, i_1 is at least one greedy neighbor of i for j and i_k is a greedy neighbor of j for i. From now on, we will use the notation "on the spanning tree" for routing decisions that lie on the embedded spanning tree. On the contrary, we denote as shortcuts, the links of the graph that do not lie on the spanning tree. An example of this embedding in the Poincaré Disc is depicted in Fig. 3(c). It is important to mention that, by construction, the root of the tree is closer in hyperbolic distance to the nodes of each subtree than the nodes lying on other subtrees.

In this work, we consider a greedy embedding of the network following [12] and we impose routing constraints on the backpressure algorithm, by determining as possible next-hop neighbors for a specific destination only "greedy" neighbors i.e. those that strictly reduce the hyperbolic distance to the destination.

III. System Model and Capacity Region

In this section we describe the system model which consists of a composition of a wireless ad hoc network and an overlay social network. This is a complex network, very likely to exist in places such as a shopping center, a conference center or a university library. The wireless ad hoc network serves the physical and network layer communications among the wireless nodes while the overlay social network represents the social interactions developed among the users and their characteristics. The graphs of the two networks are not identical as the wireless network is limited by spatial constraints whereas the social network graph is a relational graph without physical boundaries.

To begin, let us define the wireless network topology. We consider slotted time t and a wireless multi-hop network with $N(t)$ nodes at each time slot t. Each node communicates directly with all other nodes that are located inside its communication range and let $\mathcal{N}(i)$ to be the one-hop neighborhood of node i. We consider the case where $N(t)$ is constant (static network) and the case of node churn where existing nodes can leave and new nodes can join the network. The number of packets that arrive in node i for destination d at time t is $A_i^d(t)$ with finite average value $E(A_i^d(t)) = \lambda_i^d$. We suppose that each node i stores a queue $q_i^d(t)$ for each destination d. We denote with $\mu_{ij}(t)$ the communication traffic between neighboring nodes i, j at time t. Also, we denote with $\mu_{ij}^d(t)$ the limited communication traffic on the link (i, j) only for destination d. Therefore, $\sum_d \mu_{ij}^d(t) \leq \mu_{ij}(t)$. The arrival and service rates are considered bounded i.e. at each time t, we have that $\sum_d \sum_j \mu_{ij}^d(t) \leq \mu_{i,max}^{out}$, $\sum_d \sum_j \mu_{ji}^d(t) \leq \mu_{i,max}^{in}$ and $\sum_d A_i^d(t) \leq A_{i,max}$. We denote with

$dist_H(i,d)$ the hyperbolic distance between nodes i,d, as defined in Eq. (1). Finally, we use the term $I_{S(t)}$ to refer to the set of service rates of all possible independent sets of the graph, i.e. maximal sets of links that do not interfere with each other. $I_{S(t)}$ is a constant set for all t if the network is static and the channel conditions do not change.

We adapt the capacity region of [15], so as to include the routing constraints of the proposed algorithms. Therefore, the capacity region should allow routing only through greedy paths. The capacity region Λ_G is the set of all input rate matrices (λ_i^d) with $\lambda_i^d \neq 0$ if $i \neq d$ and i,d is a source-destination pair, such that there exists a rate matrix $[\mu_{ij}]$ satisfying the following constraints:

- Efficiency constraints: $\mu_{ij}^d \geq 0$, $\mu_{ii}^d = 0$, $\mu_{dj}^d = 0$, $\sum_d \mu_{ij}^d \leq \mu_{ij}$, $\forall\ i,j,d:\ i \neq d$
- Flow constraints: $\lambda_i^d + \sum_l \mu_{li}^d \leq \sum_l \mu_{il}^d, \forall\ i,d:\ i \neq d$
- Routing constraints for the Greedy backpressure: $\mu_{ij}^d = 0$ if i has at least one greedy neighbor for d and j is not one of the $i's$ greedy neighbors;
  and for Greediest backpressure: $\mu_{ij}^d = 0$ if i has at least one greedy neighbor for d and j is not $i's$ greedy neighbor that reduces at most the distance to the destination.

At this point, we define the notion of strong stability of the queues, which will be used in the proofs that follow. According to Definition 3.1 of [15], a queue, q_i^d is strongly stable if $\limsup_{t->\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} E(q_i^d(\tau)) < \infty$. If all the queues of the network are strongly stable, then the whole network is strongly stable.

In the sequel we define the overlay social network which contains information about the social interactions among the network nodes and their characteristics. We suppose that the social interactions are represented by a matrix $W_S = w_S(i,j)$ where $w_S(i,j) = 0$ if i is not a source of messages for j, i.e. if i produces messages with destination j then $w_S(i,j) > 0$. These weights, $w_S(i,j)$, admit a large variety of interpretations in social context, i.e. they may correspond to the necessity of communication between i,j, or to the mutual trust value for the i,j communication etc. In general, they express a measure of social distance for each i,j pair as described in [3]. In this paper, we translate the social weights $w_S(i,j) > 0$ as the priority or "necessity" of communication of the multi-hop connected in physical layer pair i,j. Therefore, the weights on the social graph, determine a preference on how fast each flow needs to be served. As an example, messages sent through live messenger are of higher priority than emails. Indeed, the priorities are given to the flows from a specific range of values depending on the type of the flow. These priorities will be used in the scheduling and routing algorithms defined in the following sections.

IV. Static Networks

In this section, we develop our algorithms in the case of static wireless networks, i.e. the number of nodes and their positions are fixed. As aforementioned, we propose two algorithms, the Greedy backpressure and the Greediest backpressure which both use greedy routing in hyperbolic space and backpressure scheduling. However, the Greedy backpressure defines as possible next-hop neighbors for a specific destination, all those directly connected nodes that strictly reduce the hyperbolic distance towards the particular destination, while the Greediest backpressure picks the one direct neighbor which is closest in hyperbolic distance to the destination. Therefore, the Greedy backpressure performs dynamic routing and exploits multi-path diversity, while the Greediest backpressure fixes the paths followed by the flows. The proposed algorithms differ from the classic backpressure algorithm in the definition of the weight $P_{ij}(t)$ for each link (i,j), due to their greedy routing constraints. Algorithm 1 describes how the pure backpressure algorithm is modified to follow only greedy paths.

Algorithm 1: Greedy (Greediest) backpressure algorithm for static networks, performed every time t

1 **for** *each directed link* (i,j) **do**
2 **for** *each destination* d **do**
3 *%Greedy backpressure%*
4 **if** $dist_H(i,d) > dist_H(j,d)$ **then**
5 $P_{ij}^d(t) = q_i^d(t) - q_j^d(t)$;
6 *(%OR Greediest backpressure%*
7 **if** $dist_H(i,d) > dist_H(j,d)$ *and* $dist_H(j,d) = \min_{l \in \mathcal{N}(i)} dist_H(l,d)$ **then**
8 $P_{ij}^d(t) = q_i^d(t) - q_j^d(t);$)
9 **else**
10 $P_{ij}^d(t) = -\infty$;
11 *%Define the weight* $P_{ij}(t)$ *as follows*:
12 $P_{ij}(t) = \max(\max_d P_{ij}^d(t), 0)$;
13 $d^*(i,j) = arg\max_d P_{ij}^d(t)$;
14 *Choose the rate matrix through the maximization*:
15 $[\mu_{ij}(t)] = arg\max_{\mu \in \mathbf{I_{S(t)}}} \sum_{(i,j)} \mu_{ij} P_{ij}(t)$
16 **for** *each directed link* (i,j) **do**
17 **if** $\mu_{ij}(t) > 0$ **then**
18 *the link* (i,j) *serves* $d^*(i,j)$ *with* $\mu_{ij}^{d^*}(t) = \mu_{ij}(t)$;
19 *For* $d \neq d^*$ *we set* $\mu_{ij}^d(t) = 0$

It is important to mention that due to the greedy embedding that ensures the existence of a greedy path for every source-destination pair, the routing constraints of Algorithm 1 are well defined and there do not exist local minima that can cause the packets to get stack at a specific node. Therefore, with probability one the packets will be routed to the destination under Algorithm 1.

In the sequel, we will use the flow priorities in order to break ties in two cases. The first case takes place when an equality in the difference of the queue backlogs for two or more destinations exists on a link; in line 12 of Algorithm 1. The second case refers to line 15 of Algorithm 1, that chooses the independent set $I(t) \in I_{S(t)}$, that achieves the maximum $\sum_{(i,j) \in I(t)} \mu_{ij}(t) P_{ij}(t)$. For both cases, we suppose that each link can serve only one packet, at a time slot t. For the first

case, when comparing between two queue differences, we break ties as follows:

if $q_i^{d_1}(t) - q_j^{d_1}(t) = q_i^{d_2}(t) - q_j^{d_2}(t)$ *for* d_1, d_2 **then**
  if *the weight of the first packet in* $q_i^{d_1}(t)$ *is larger than the corresponding one in* $q_i^{d_2}(t)$ **then**
    $d^* = d_1$;
  if *the weight of the first packet in* $q_i^{d_1}(t)$ *is equal to the corresponding one in* $q_i^{d_2}(t)$ **then**
    d^* *becomes equal to either* d_1 *or* d_2 *with equal probability*;
  else
    $d^* = d_2$;

After, assigning a weight $P_{ij}(t)$ at each link (i, j), we also assign a priority weight to the link (i, j), $w_p(i, j, t)$, which is equal to the weight of the flow (packet) that is going to be served. For the second case, when comparing between the sums of line 15 for two independent sets in order to choose the independent set $I(t)$ that maximizes this sum, we break ties as follows:

if $\sum_{(i,j)\in I_1} \mu_{ij} P_{ij}(t) = \sum_{(i,j)\in I_2} \mu_{ij} P_{ij}(t)$ **then**
  if $\sum_{(i,j)\in I_1} w_p(i, j, t) > \sum_{(i,j)\in I_2} w_p(i, j, t)$ **then**
    $I(t) = I_1$;
  if $\sum_{(i,j)\in I_1} w_p(i, j, t) = \sum_{(i,j)\in I_2} w_p(i, j, t)$ **then**
    $I(t)$ *becomes equal to either* I_1 *or* I_2 *with equal probability*;
  else
    $I(t) = I_2$;

In the case that a link transmits more than one packet at a time slot, the above algorithms can be modified by considering the average weight or the highest weight or the middle weight of the packets that are going to be transmitted. In addition, by ignoring the flow priorities, we are led to the standard Greedy and Greediest backpressure algorithms which do not take into consideration the social weights.

The following theorem shows the throughput optimality of Algorithm 1 for static wireless multi-hop networks.

Theorem 1: If we assume that the arrival rates λ_i^d lie inside the capacity region Λ_G, then the queues of the network are strongly stable, under the Greedy (Greediest) backpressure algorithm for static networks.

Proof: We define two indicator functions dependent on the type of the backpressure algorithm (Greedy or Greediest).

For the Greedy backpressure:

$I_1 = \{dist_H(i,d) > dist_H(j,d) \bigwedge (j \in \mathcal{N}(i))\}$, $I_2 = \{dist_H(i,d) < dist_H(j,d) \bigwedge (i \in \mathcal{N}(j))\}$,

while for the Greediest backpressure:

$I_1 = \{dist_H(i,d) > dist_H(j,d) \bigwedge dist_H(j,d) = \min_{l\in\mathcal{N}(i)} dist_H(l,d) \bigwedge (j \in \mathcal{N}(i))\}$

$I_2 = \{dist_H(i,d) < dist_H(j,d) \bigwedge dist_H(i,d) = \min_{l\in\mathcal{N}(j)} dist_H(l,d) \bigwedge (i \in \mathcal{N}(j))\}$

where we observe that I_2 equals I_1 if we replace i, j with j, i correspondingly.

The queue dynamics in the case of Algorithm 1 are

$$q_i^d(t+1) = \max\{q_i^d(t) - \sum_{j|I_1} \mu_{ij}^d(t), 0\} + \sum_{j|I_2} \mu_{ji}^d(t) + A_i^d(t). \quad (2)$$

We denote by $Q(t) = (q_i^d(t))$, the vector of queues of the network. We define the Lyapunov function $L(Q(t)) = \sum_{i,d} q_i^d(t)^2$ and take the expected value of the difference $L(Q(t+1)) - L(Q(t))$ so as to compute the Lyapunov Drift:

$$\begin{aligned}
&E(L(Q(t+1)) - L(Q(t))|Q(t)) = E\left[\sum_{i,d} q_i^d(t+1)^2 - \sum_{i,d} q_i^d(t)^2 \ \Big|Q(t)\right],\\
&\overset{(2),[15]}{\leq} \sum_{i,d}\left\{q_i^d(t)^2 + E\left[\left(\sum_{j|I_1}\mu_{ij}^d(t)\right)^2 + \left(\sum_{j|I_2}\mu_{ji}^d(t) + A_i^d(t)\right)^2 \ \Big|Q(t)\right]\right\}\\
&- \sum_{i,d}\left\{2q_i^d(t)E\left[\left(\sum_{j|I_1}\mu_{ij}^d(t) - \sum_{j|I_2}\mu_{ji}^d(t) - A_i^d(t)\right)\Big|Q(t)\right]\right\} - \sum_{i,d} q_i^d(t)^2,\\
&\leq B + 2\sum_{i,d} q_i^d(t)\lambda_i^d - 2\sum_{i,d} q_i^d(t)E\left[\left(\sum_{j|I_1}\mu_{ij}^d(t) - \sum_{j|I_2}\mu_{ji}^d(t)\right)\Big|Q(t)\right], \quad (3)
\end{aligned}$$

where $\mu_{ij}^d(t)$ are the service rates computed by Algorithm 1 and $B > 0$ is an upper bound of $\sum_{i,d} E\left[\left(\sum_{j|I_1}\mu_{ij}^d(t)\right)^2 + \left(\sum_{j|I_2}\mu_{ji}^d(t) + A_i^d(t)\right)^2 \Big|Q(t)\right]$.

If the λ_i^d lie inside the capacity region, then from Corollary 3.9 in [15], there exist rates $\hat{\mu}_{ij}^d(t)$ determined according to the network topology and independently of the queue backlog satisfying $\lambda_i^d + \epsilon = E\left[\sum_{j|I_1}\hat{\mu}_{ij}^d(t) - \sum_{j|I_2}\hat{\mu}_{ji}^d(t)\right]$ $\forall$ i, d, $\epsilon > 0$.

The Greedy (Greediest) backpressure maximizes $\sum_{i,d} q_i^d(t) E\left[\left(\sum_{j|I_1}\mu_{ij}^d(t) - \sum_{j|I_2}\mu_{ji}^d(t)\right)\Big|Q(t)\right]$, so

$$\sum_{i,d} q_i^d(t) E\left[\sum_{j|I_1}\mu_{ij}^d(t) - \sum_{j|I_2}\mu_{ji}^d(t)\Big|Q(t)\right] > \sum_{i,d} q_i^d(t) E\left[\sum_{j|I_1}\hat{\mu}_{ij}^d(t) - \sum_{j|I_2}\hat{\mu}_{ji}^d(t)\right]. \quad (4)$$

Therefore the Lyapunov drift (Eq. (3)) becomes

$$E[L(Q(t+1)) - L(Q(t))|Q(t)] \leq B - 2\sum_d \sum_i q_i^d(t)\epsilon, \quad (5)$$

and from lemma 4.1 of [15], the network is strongly stable. ■

V. Impact of the Hyperbolic Embedding on the Delay Performance of the Greedy (Greediest) Backpressure

In this section we study how the choice of the spanning tree, used for the greedy hyperbolic embedding, can influence the delay performance of the proposed algorithms. The delay performance of a scheduling/routing algorithm for each flow, is closely related to the sum of queues of the nodes lying on the source-destination path. As stated in [10] the total backlog of a path increases with the increase of its hop-length. More precisely, according to [10], in the case of the backpressure scheduling (with fixed routing), the queue backlog at each node lying on the path of a flow increases as we move from the destination node to the source node, where the destination node has zero backlog. In addition to this, Theorem 1 of [10], states that under the backpressure

scheduling, the steady-state expected value of the sum of queue lengths along the route of any flow f is bounded by a quantity proportional to $\frac{|F|K_{\max}^2}{\lambda_f}$ where $|F|$ is the number of flows in the network, λ_f is the input rate of flow f and $K_{\max}$ is the maximum number of hops of all the flows in the network. In the sequel, we prove that in the case of Greedy (Greediest) routing, $K_{\max}$ can be upper-bounded, by a quantity which depends on spanning tree parameters (depth and number of subtrees), which suggests the intuition that the delay performance can be spanning tree dependent.

We denote with L_f the set of leaves (nodes with degree 1) of the spanning tree and with $|L_f|$, its cardinality. Also, we symbolize with D, the depth of the spanning tree.

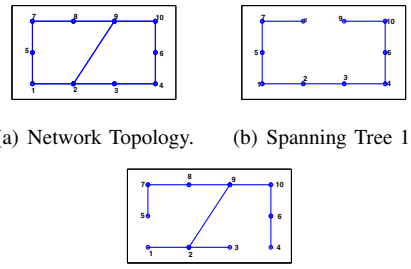
Proposition 1: By using the greedy hyperbolic embedding of [12] and under the Greedy (Greediest) routing for static networks, $K_{\max}$ can be upper bounded by:

$$K_{\max} \leq (D-1)\left(|L_f|-1+\mathbf{1}_{L_f\geq 3}\binom{|L_f|-1}{2}\right)+2D$$

where $\mathbf{1}$ is the indicator function.

Proof: A node willing to deliver a packet to a particular destination according to the Greedy (Greediest) routing, has two possible choices. The first one is to forward the packet to its parent or one of its children on the spanning tree, depending on which of them satisfies the Greedy (Greediest) routing constraints (there is one of them satisfying the Greedy routing constraints due to the greedy embedding). The second choice is to follow a shortcut that satisfies the Greedy (Greediest) routing constraints. For the Greediest backpressure only one of these two choices will be viable as the sender chooses the neighbor which reduces at most the distance to the destination (supposing that the distances are unique). The shortcuts can be divided in two categories: the intra-subtree ones, which link nodes on the same subtree, and the inter-subtree ones, which connect different subtrees. The maximum number of inter-subtree shortcuts, can be computed using the maximum number of subtrees. We need to note here that a packet which reaches to the subtree of the destination remains on it, otherwise it will increase its hyperbolic distance to the destination violating the greedy routing constraints (Section II). Therefore, the subtree of the destination is like an absorbing state, i.e. there is no shortcut directed from this subtree to any other subtree.

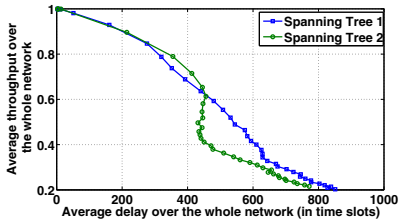
The maximum possible number of subtrees is equal to the number of leaves of the spanning tree, i.e. $|L_f|$. Therefore the maximum number of inter-subtree shortcuts, denoted as $|S|$, is $|S| = |L_f|-1+\mathbf{1}_{L_f\geq 3}\binom{|L_f|-1}{2}$, where the first two summands correspond to the shortcuts of all other subtrees to the subtree including the destination and the third summand corresponds to the total number of shortcuts between any two subtrees except the one of the destination. If $|L_f|<3$, only shortcuts on the same subtree or towards the subtree of the destination are allowed (both of which reduce the hop-length of the routing path). In addition, after an inter-subtree shortcut, a packet can move up to $D-1$ hops on the spanning tree (or by following intra-subtree shortcuts) towards the root, before being transferred by another inter-subtree shortcut. Finally, the greedy path lying on the spanning tree can be at most $2D$ hops in length. By computing $|S|(D-1)+2D$, we arrive to the desirable result. ■



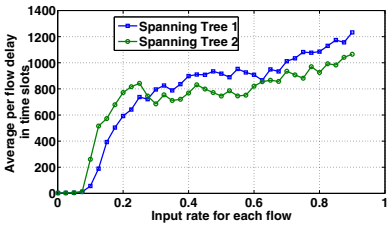
(a) Network Topology. (b) Spanning Tree 1.

(c) Spanning Tree 2.

Fig. 1. Two different spanning trees chosen for the network topology.



(a) Delay-Throughput Trade-off.



(b) Delay vs. Input Rate.

Fig. 2. Performance curves for the same network (Fig. 1(a)) with different spanning trees used for the hyperbolic embedding (Fig. 1(b), 1(c)).

We have performed many MATLAB simulations which show that a different choice of spanning tree for the hyperbolic embedding may lead to different throughput-delay curves and different average delays experienced by the flows. Fig.1 and Fig. 2 show representative ones.

VI. Dynamic Networks

In dynamic conditions, the topology of the network (and the capacity region) changes due to the addition and deletion of nodes. We assume that the network remains always connected and that nodes come and leave in a much slower rate than the rate of the scheduling/routing process. In this case, new nodes can be embedded in the hyperbolic space according to the greedy embedding of [12]. In this section, we suppose that new nodes can enter and existing nodes can leave the network and that there is an upper bound on the number of nodes, $N_{\max}$, possible to be simultaneously in the network, which is realistic for wireless networks. Therefore the number of nodes is finite at all times. As a result, the lines $4-5$, $9-10$ of Algorithm 1 for the Greedy backpressure are replaced with the following ones:

if *i has at least one greedy neighbor for a destination d* **then**
  if $dist_H(i,d) > dist_H(j,d)$ **then**
    $P_{ij}^d(t) = q_i^d(t) - q_j^d(t)$;
  else
    $P_{ij}^d(t) = -\infty$;

if *i has not greedy neighbor for a destination d* **then**
  $P_{ij}^d(t) = q_i^d(t) - q_j^d(t)$;

For the Greediest backpressure, the lines $7-8$, $9-10$ change similarly as:
if *i has at least one greedy neighbor for a destination d* **then**
  if $dist_H(i,d) > dist_H(j,d)$ *and* $dist_H(j,d) = \min_{l\in\mathcal{N}(i)} dist_H(l,d)$ **then**
    $P_{ij}^d(t) = q_i^d(t) - q_j^d(t)$;
  else
    $P_{ij}^d(t) = -\infty$;

if *i has not greedy neighbor for a destination d* **then**
  $P_{ij}^d(t) = q_i^d(t) - q_j^d(t)$;

As in section IV, we can use the priority weights $w_S(i,j)$ of the flows to break ties in the computation of the weights $P_{ij}(t)$ of the links and the selection of the independent set $I(t) \in I_{S(t)}$ that is going to transmit.

The following theorem shows under certain assumptions the throughput optimality of the Greedy (Greediest) backpressure for dynamic networks. In order to prove Theorem 2, we made three main assumptions. Firstly, we consider that at each time instant only one node can be added or deleted. Secondly, a new node does not change any pre-existing connections in the network. Thirdly, there is a controller that adapts the arrival rates to lie inside the capacity region, if the network topology changes.

Theorem 2: Assuming that at each time interval T_i with constant number of nodes $N(T_i)$, the arrival rates, which are dependent on T_i, $\lambda_i^d(T_i)$, lie inside the capacity region depending on T_i, $\Lambda_G(T_i)$, then the queues are strongly stable.

Proof: We only provide the proof sketch due to space limitations. Firstly, we prove that for each interval T_i with constant number of nodes $N(T_i)$, if the arrival rates are adapted through a controller to lie inside the capacity region of this interval $\Lambda_G(T_i)$, the sum of queues is bounded (when it is over a specific value the drift becomes negative so as to reduce it). This proof is similar to the corresponding one in Theorem 1. Next we check the Lyapunov drift at the transitions between two intervals with different number of nodes and since we have assumed that at each transition only one node can be added or deleted, two consecutive intervals differ only by one node. Omitting some further tedious details we conclude the strong stability of the queues. ■

VII. Incorporating Trust in the Greedy Backpressure Algorithm

In this section, we propose an extension of the Greedy backpressure algorithm in order to take into consideration possible weights on the links of the physical graph, in addition to distance and congestion gradients. A weighted network graph contains much more information for the communicating node pairs, compared with the adjacency matrix, such as the opinion that each node has for each one of its neighbors, expressed as trust value, or the quality of cooperation between two nodes expressed as mutual trust value, or possible delay experienced due to the medium, etc. This information determines the collaboration for communication among the selfish nodes of a social network. In this work we focus on weights representing mutual trust values between neighboring nodes, but it can be easily translated to weights that represent cost of communication or channel delays. Towards this direction, we assume that the physical layer graph is weighted with values expressing mutual trust between neighboring nodes, showing the quality of the relationship between them.

Let us denote with $W = [w(i,j)]$, the symmetric matrix containing the weights of the physical graph. The trust value of a path p connecting a pair of nodes is given by the product of the trust values of the links lying on the path, i.e. $TV(i,j,p) = \Pi_{(l,k)\in p} w(l,k) = TV(j,i,p)$. The trust value $TV(i,j)$ of a pair of nodes i,j is the maximum $TV(i,j,p)$ for all paths p. If we replace every weight on the graph $w(i,j)$ with $-log(w(i,j))$, we can transform the problem of finding the most trusted path between two nodes to a shortest path problem, where the new weights, $-log(w(i,j))$, are considered as costs of communication. Also, we know that given a weighted graph G on N vertices, one can get a natural metric d_G by setting, for every $i,j \in V(G)$, the distance $d_G(i,j)$ to be the length of the shortest-path between vertices i and j in G.

In the proposed Greedy backpressure algorithm we choose a random spanning tree on the initial graph, embed it in the hyperbolic space and we route packets only through greedy paths. In the case where we want to enhance our algorithm to take into consideration the quality of communication between two nodes, we can simply choose the most trusted greedy path. However, there is the risk of choosing only low trusted paths for a pair of nodes i,j, due to the fact that the trust values of the available greedy paths might be far less than the trust value $TV(i,j)$ of the most trusted path in the network (which might not be greedy). Therefore, even if, for the i,j communication, we choose the greedy path with the highest trust value, this might be far less than $TV(i,j)$. In order to improve the trustworthiness of the greedy paths, we propose to use the algorithm of [16] for the spanning tree construction, which will be used for the embedding. This algorithm transforms a graph metric into a spanning tree metric for weighted graphs. In the graph metric, the shortest distance between two nodes is considered over all the links of the network. In contrast, in the spanning tree

metric there is a unique path on the tree connecting two nodes which also corresponds to their shortest distance. In order to transform a graph metric d_G into a spanning tree metric, we consider probabilistic embeddings into spanning tree metrics. A probabilistic embedding into spanning trees is a probability distribution over the spanning trees T of the graph G. Let us denote with d_t the shortest path between a pair of nodes in the t spanning tree graph. The quality of a probabilistic embedding is given by the expected distortion.

$$\text{Expected Distortion} = E_{all\ t\in T}\left(\frac{d_t(ij)}{d_G(ij)}\right) \tag{6}$$

We use the algorithm of [16] to construct one tree of the distribution T with expected stretch on the length of each edge (i,j) equal to $O(logn(loglog(n))^2 logloglogn)$. In the sequel, we embed the constructed spanning tree on the hyperbolic space using Crovella's aglorithm [12].

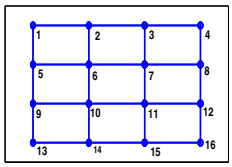
Although by optimizing over all greedy paths to obtain the one with the lower cost value, or higher trust value, ensures a lower bound of the trust value of the communicating path, it is computationally complex for a wireless ad hoc network. We now propose a distributed implementation with the same expected lower bound of achieved trust. Let us denote with $TV(i,j,sp)$ the trust value achieved between i,j on the unique greedy path sp lying on the spanning tree and with $TV(i,j,p_h)$ the corresponding trust value of the path through shortcut h.

if *i transmits to j* **then**
  Computes for all shortcuts,h, to j the value :
  $TV(i,j,p_h) = TV(i,h,(i,h)) + TV(h,j,sp)$;
  Chooses $\max(\max_h(TV(i,j,p_h)), TV(i,j,sp))$;
  %Forwards to the next hop neighbor, according to the TV selected%
  if $TV(i,j,p_h)$ *for a h is selected* **then**
    Forwards to this h through the shortcut (i,h);
  else
    Forwards to its unique greedy neighbor for j on the spanning tree;

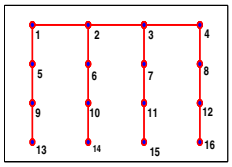
Therefore, the node i uses a shortcut only if it can improve the trust value $TV(i,j,sp)$. This algorithm achieves the same lower bound of trust value of the chosen path as the optimization method, which is equal to the trust value of the greedy path lying on the spanning tree.

VIII. Simulation Results

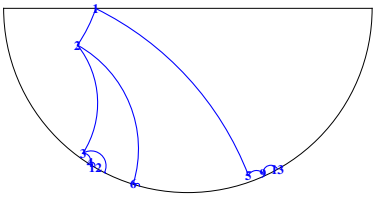
In this section, we evaluate the performance of our algorithms through MATLAB simulations. The physical network consists of a 4x4 grid topology, as shown in Fig. 3(a), where each node at each time slot generates traffic for a random destination, with probability equal for all node pairs ranging from $\lambda = 0.025$ to $\lambda = 0.5$ with step increase 0.025. Therefore, the social network is the complete 16−node graph. The priority weights of the social flows take two discrete values $\{1,2\}$, i.e. we consider flows of two different priorities in the social overlay network. The flows of priority 2 prefer to be served faster, therefore, they are selected to break ties, as described in Section IV. Fig. 3(b) depicts the shortest path spanning tree rooted at the node with ID 1, used for the greedy embedding of the graph in the hyperbolic space. In Fig. 3(c), the hyperbolic embedding of the tree in Fig. 3(b) is illustrated. The non-tree (not embedded edges) are not shown in Fig. 3(b), 3(c).



(a) Grid Network Topology.



(b) Shortest path spanning tree routed at node 1.



(c) Hyperbolic Embedding of the spanning tree.

Fig. 3. Topology and Spanning Tree.

For each probability λ we run the algorithms for 5000 slots. Each link can transmit one packet at a time slot. The one-hop interference model determines the independent sets of the graph i.e. the links that are directly connected through a common neighbor cannot transmit simultaneously. In Fig. 4, throughput is expressed as the percentage of packets that reach their destination divided by those sent from the source for each flow and both throughput and delay are expressed as averages for all flows (source-destination pairs). The static network embedded in the hyperbolic space runs the Greedy, Greediest and the original backpressure algorithms correspondingly. The bullets on the curves represent different values of λ. The simulation results demonstrate the Pareto dominance of our algorithms over the pure backpressure algorithm, concerning the throughput-delay trade-off. The proposed algorithms, achieve lower delays for the same values of throughput compared with the backpressure algorithm. The Greediest backpressure achieves the best delay-throughput trade-off for the corresponding communication traffic pattern and static physical topology.

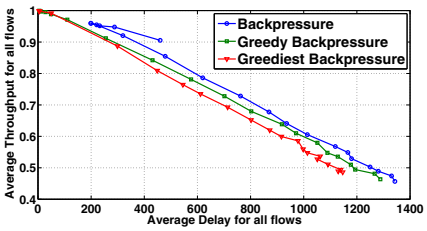


Fig. 4. Static Networks: Evaluation of Throughput-Delay Performance of the proposed algorithms. Throughput and Delay averaged per flow.

In order to study the performance of the proposed algo-

rithms under dynamic network conditions, we formulated the following scenario. The physical and social layer network graphs remain the same as in the case of the static networks above. For each λ we run the algorithms for 5000 slots. Every 1000 time slots we delete one node from the topology online with the below order, we first delete the node with ID 6 and then the nodes with IDs $8, 10, 15$. Due to these deletions the greedy property is locally lost and the network runs the dynamic version of Greedy (Greediest) backpressure. The simulation results are shown in Fig. 5 where the notations are similar as in Fig. 4 for static networks. We observe that for all λ, the Greedy and Greediest backpressure algorithms achieve a better throughput delay trade-off than the classic backpressure algorithm, as for the same values of throughput they lead to lower delays.

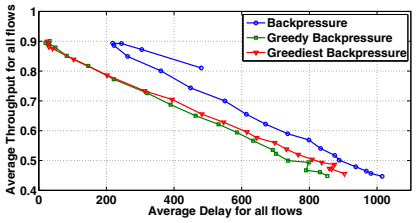


Fig. 5. Dynamic Networks: Evaluation of Throughput-Delay Performance of the proposed algorithms. Throughput and Delay averaged per flow.

In order to check how the prioritized break ties achieves a better delay of the highest priority flows, we present the average delay of the flows with priority 1, 2 and the average delay over all flows in the case of the Greedy backpressure algorithm. Similar results were obtained for the other algorithms.

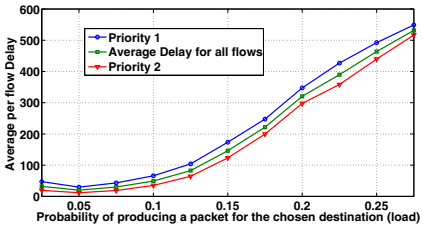


Fig. 6. Comparison of the delay for flows with different priorities.

We observe that the flows with higher priority weight (equal to 2) achieve delays lower than the total average delay value and lower than the delays experienced by flows of lower priority (priority weight equal to 1), especially for lower and average input traffic rates. However we do not expect very large differences as the prioritization of the flows is limited only in the case when we have to break ties.

IX. Conclusions

In this paper, we proposed, analyzed and evaluated algorithms for social-wireless networks based on hyperbolic embedding, greedy routing and backpressure scheduling. We proved their throughput optimality and showed through simulations their dominance over the pure backpressure algorithm concerning the throughput-delay trade-off. Also, the proposed algorithms are social-aware and use information from the social layer graph to break ties. Finally, we studied the impact of the spanning tree selection for the delay performance and extended the proposed algorithms in the case of weighted physical layer graphs. In future work, we are planning to compare the Greedy (Greediest) backpressure algorithm with other algorithms existing in related work for improving the pure backpressure algorithm. However, the proposed algorithms in this work use both congestion gradients as the classic backpressure and distance gradients, through a distributed implementation with only local information and without additional complexity.

Acknowledgments

The research of Eleni Stai and John Baras was partially supported by the US Air Force Office of Scientific Research under MURI grants FA9550-10-1-0573 and FA9550-09-1-0538, and by the National Science Foundation under grant CNS-1018346. Eleni Stai was also partially supported by the Foundation for Education and European Culture (IPEP).

References

[1] E. Sarigol, O. Riva, P. Stuedi, G. Alonso, "Enabling Social Networking in Ad Hoc Networks of Mobile Phones," *In Proc. of the VLDB Endowment*, Vol. 2, No. 2, pp. 1634 - 1637, August 2009.

[2] L. Juan, S. U. Khan, "MobiSN: Semantics-Based Mobile Ad Hoc Social Network Framework," *In Proc. of the IEEE Global Telecommunications Conference (GLOBECOM)*, 2009.

[3] V. Kulkarni, M. Devetsikiotis, "Social Distance Aware Resource Allocation in Wireless Networks," *In Proc. of the IEEE Global Telecommunications Conference (GLOBECOM)*, 2009.

[4] E. Stai, V. Karyotis and S. Papavassiliou, "Topology Enhancements in Wireless Multi-hop Networks: A Top-down Approach," *IEEE Trans. on Parall. & Distr. Sys.*, Vol. 23, No. 7, pp. 1344 - 1357, 2012.

[5] Y. Chen, C. Caramanis, S. Shakkottai, "On File Sharing over a Wireless Social Network," *In Proc. of the IEEE Int'l Symposium on Information Theory (ISIT)*, 2011.

[6] B. Azimdoost, H. R. Sadjadpour, J. J. Garcia-Luna-Aceves, "Capacity of Composite Networks: Combining Social and Wireless Ad Hoc networks," *In Proc. of the IEEE Wireless Communications and Networking Conference (WCNC)*, 2011.

[7] L. Tassiulas, A. Ephremides, "Stability Properties of Constrained Queueing Systems and Scheduling Policies for Maximum Throughput in Multihop Radio Networks," *IEEE Transactions on Automatic Control*, Vol. 37, No. 12, pp. 1936 - 1948, December 1992.

[8] A. Eryilmaz, R. Srikant, "Joint Congestion Control, Routing, and MAC for Stability and Fairness in Wireless Networks," *IEEE J. on Selected Areas in Com. (JSAC)*, Vol. 24, No. 8, pp. 1514 - 1524, July 2006.

[9] L. Ying, S. Shakkottai, A. Reddy, S. Liu, "On Combining Shortest-Path and Back-Pressure Routing Over Multi-hop Wireless Networks," *IEEE/ACM Tran. on Net.*, Vol. 19, No. 3, pp. 841 - 854, 2011.

[10] L. Bui, R. Srikant, A. L. Stolyar, "Novel Architectures and Algorithms for Delay Reduction in Back-pressure Scheduling and Routing," *In Proc. of the IEEE INFOCOM*, pp. 2936 - 2940, April 2009.

[11] H. Xiong, R. Li, A. Eryilmaz, E. Ekici, "Delay-Aware Cross-Layer Design for Network Utility Maximization in Multi-Hop Networks," *IEEE J. on Selected Areas in Com. (JSAC)*, Vol. 29, No. 5, pp. 951 - 959, May 2011.

[12] A. Cvetkovski, M. Crovella, "Hyperbolic Embedding and Routing for Dynamic Graphs," *IEEE INFOCOM*, pp. 1647 - 1655, April 2009.

[13] R. Kleinberg, "Geographic Routing Using Hyperbolic Space," *In Proc. of the IEEE INFOCOM*, pp. 1902 - 1909, May 2007.

[14] J. W. Anderson, "Hyperbolic Geometry," *2nd ed. Springer*, 2007.

[15] L. Georgiadis, M. J. Neely, L. Tassiulas, "Resource Allocation and Cross-Layer Control in Wireless Networks," *NOW publications*, 2006.

[16] I. Abrahaam, Y. Bartal, O. Neiman, "Nealry Tight Low Strech Spanning Trees," *In Proc. of the 49th Annual IEEE Symposium on Foundations of Computer Science*, 2008.