Flow Optimization in Delay Tolerant Networks using Dual Decomposition

Savvas Gitzenis, George Konidaris, Stavros Toumpis

Flow Optimization in Delay Tolerant Networks using Dual Decomposition

Savvas Gitzenis
Informatics & Telematics Institute, CERTH, Greece

George Konidaris and Stavros Toumpis
Informatics Department, AUEB, Greece

***Abstract*—We study flow optimization in Delay Tolerant Networks (DTNs), which we model using *Capacity Region Evolving Graphs (CREGs)*. CREGs consist of different instances (called *replicas*) of the network graph in cascade; each replica is associated with a distinct time period (called *epoch*) and its own Capacity Region. Although CREGs can model any DTN, they are particularly well suited for the study of wireless ones. We define a single-commodity utility maximization problem in a CREG of T replicas that contains as special cases various interesting flow maximization problems. Using dual decomposition, we cast the maximization as a dual problem that can be solved iteratively and where in each iteration a set of T problems, T times smaller than the original, are solved, potentially (if multiple processors are available) *in parallel*. In addition, we propose two suboptimal utility maximization heuristics that operate on an epoch-by-epoch basis and we discuss a multi-commodity extension to the problem.**

I. Introduction

We study Delay Tolerant Networks (DTNs), where very large delays in the delivery of data, comparable with the time it takes the topology to change appreciably, are either unavoidable or acceptable [1]. In the last few years, a large number of routing protocols have been devised for use in DTNs [2], [3], [4]. Complementing the design of protocols, research on optimal routing, in the graph theoretic sense, has been undertaken, using evolving graphs [5] and related tools developed in the context of dynamic flows [6], [7].

An important class of DTNs are Store-Carry-and-Forward wireless networks, where data arrive at their destination partly through wireless forwarding and partly through physical transport by intermediate nodes [8]. A striking feature of this networking paradigm is that nodes are expected to buffer transient packets for significant amounts of time, while their physical transport takes place. Thus, buffer sizes, and their efficient management, become key parameters for the performance of such networks, in contrast to the conventional wisdom in ordinary networks that buffer sizes do not matter provided that they are not too small.

In Section II, we develop a DTN model that we term the *Capacity Region Evolving Graph (CREG)*. As with the Evolving Graphs introduced in [5], CREGs capture the evolution of the network topology by discretizing time into a set of T *epochs*, each associated with its own *replica* subgraph. However, with respect to Evolving Graphs, CREGs are augmented to include additional nodes in between the replicas as well as joint constraints on the replica arc capacities in terms of capacity regions. We assume that all packets are of the same type, i.e., there is a single commodity. We define the *DTN Utility Maximization (DTNUM) Problem* as follows: we assume a convex cost incurred to each node for having a given volume of data at the start of the first epoch, and a concave utility derived by each node from having a given volume of data at the end of the last epoch. The DTNUM Problem is then the maximization of the sum of all utilities, minus the sum of all costs, given the buffer storage constraints. By appropriately selecting the utilities and costs, we can address a variety of interesting cases, such as maximizing the flow between one or more sources and one or more sinks.

In Section III we convert this problem, using a dual decomposition technique, into a dual problem that is solved iteratively. Each iteration consists of T smaller problems, each problem corresponding to a different epoch. The T problems are independent of each other, permitting their parallel process inside each iteration.

Note that, due to coupling over time, in order to find the optimal flow in *any* epoch we need to know the topology in *all* epochs, both preceding and succeeding. This is clearly restricting, however we note that there exist DTNs whose evolution is known a priori (such as DTNs where the changes in the topology are induced by periodic movements of nodes [9]). It is also helpful to know the performance of the optimal routing schedule, even if it impossible to apply it, in order to use it as a benchmark for the performance of practicable, causal routing schemes. Finally, studying the optimal non-causal routing schedules provides intuition that can be used for the efficient design of good causal ones. Indeed, in Section V two such causal schemes, motivated by our dual decomposition, are presented.

In Section VI we extend our formulation to the case where there are multiple commodities and costs associated with the use of links and the storage of data at node buffers. As we show, a dual decomposition approach is still possible.

In Section VII related work is discussed, notably in the areas of dynamic flows, utility maximization, and capacity regions. We conclude in Section VIII.

II. Flow Optimization Framework

The network is modeled as a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ consisting of a set $\mathcal{N}$ of N nodes $1, \ldots, N$, and a set $\mathcal{A}$ of A directed arcs (i, j), where $i, j \in \mathcal{N}$. Time is divided in **epochs** $t = 1, \ldots, T$.

978-1-61284-824-2/2012 - Copyright is with IFIP

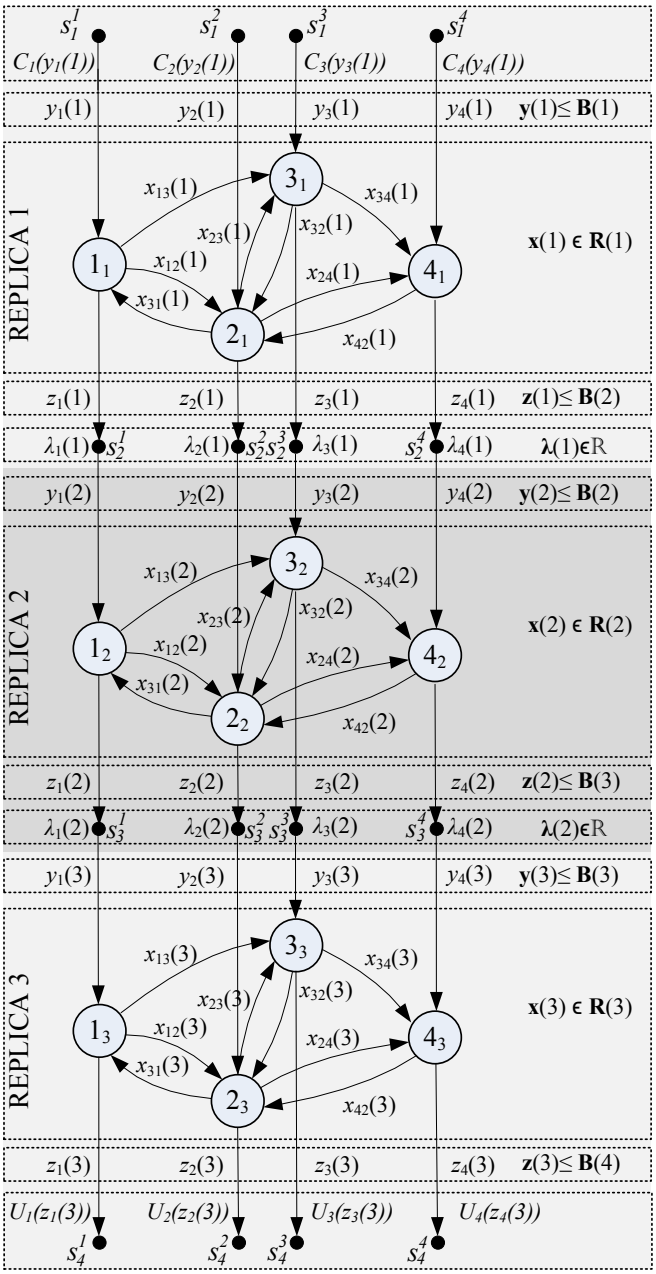

Fig. 1. An example Capacity Region Evolving Graph (CREG).

To model the storage constraints of individual nodes, we assume that the maximum amount of data that a node i can keep in its buffers during the transition between epochs $t-1$ and t, where $t = 2, \ldots, T$, is the **internal buffer size** $B_i(t)$. Define the **internal buffer size vectors** $\mathbf{B}(t) \triangleq (B_i(t) : i \in \mathcal{N}),\ t = 2, \ldots, T$. Each arc $(i,j) \in \mathcal{A}$ is associated at epoch t with a non-negative **link flow** $x_{ij}(t)$, which represents the volume of traffic (in bits per epoch) that flows through link (i,j) during the whole duration of the epoch t. Define then the **link flow vector** during epoch t, $\mathbf{x}(t) \triangleq (x_{ij}(t) : (i,j) \in \mathcal{A})$. Vector $\mathbf{x}(t)$ is required to belong to a given convex A-dimensional set called the epoch's **capacity region** $\mathbf{R}(t)$. In other words, it is possible for all links to support simultaneously the volume of traffic specified by $\mathbf{x}(t)$ iff $\mathbf{x}(t) \in \mathbf{R}(t)$. Therefore, whether a link can support a given volume of traffic or not depends on the flows in other links in the network through the requirement that they jointly belong to $\mathbf{R}(t)$. Observe that capacity regions are fixed for the duration of an epoch, and that the variations in the network topology (for example due to node movement, in wireless DTNs) are modeled through the instantaneous change, on epoch transitions, of the capacity region used.

We now create the Capacity Region Evolving Graph (see Fig. 1 for an example): first, we create T **replica** subgraphs indexed by $t = 1, \ldots, T$. Each node i of the physical network corresponds to T replica nodes $i_1, \ldots, i_T$, and each arc (i,j) corresponds to T replica arcs $(i,j)_1, \ldots, (i,j)_T$. Replica t models the network during epoch t, therefore the flow through arc $(i,j)_t$ is $x_{ij}(t)$, and the set of flows during that epoch is the link flow $\mathbf{x}(t)$. Replicas are interleaved with $T+1$ sets of **storage nodes** $\{s_t^1, s_t^2, \ldots, s_t^N\}$, for $t = 1, \ldots, T+1$, as follows: for $t = 1, \ldots, T$, node i_t is connected to storage node s_{t+1}^i through a directed arc (i_t, s_{t+1}^i) and storage node s_t^i is connected to i_t through a directed arc (s_t^i, i_t). The non-negative **input flow** through (s_t^i, i_t), coming into replica t at node i, is denoted by $y_i(t)$, and the set of these flows is denoted by the **input flow vector** $\mathbf{y}(t) \triangleq (y_i(t) : i \in \mathcal{N}),\ t = 1, \ldots, T$. Similarly, the non-negative **output flow** through (i_t, s_{t+1}^i), coming out of replica t at node i, is denoted by $z_i(t)$, and the set of these flows is denoted by the **output flow vector** $\mathbf{z}(t) \triangleq (z_i(t) : i \in \mathcal{N}),\ t = 1, \ldots, T$.

We associate the input flows $\mathbf{y}(1)$ of the *first* replica with a set of increasing, convex **input cost functions** $C_i(y_i(1))$, one for each node i. We associate the output flows of the *last* replica with a set of increasing concave **utility functions** $U_i(z_i(T))$, one for each node i. Moreover, we define the **external buffer size vectors** $\mathbf{B}(1)$, $\mathbf{B}(T+1)$ as constraints to the first input and last output flows:

$$\begin{aligned} \mathbf{y}(1) &\leq \mathbf{B}(1) \triangleq (B_1(1), \ldots, B_N(1)), \\ \mathbf{z}(T) &\leq \mathbf{B}(T+1) \triangleq (B_1(T+1), \ldots, B_N(T+1)). \end{aligned}$$

Consider the following convex optimization problem, with optimization variables $\mathbf{x}(t),\ \mathbf{y}(t),\ \mathbf{z}(t),\ t = 1, \ldots, T$.

Problem I: DTN Utility Maximization (DTNUM)

$$\text{maximize:} \quad \sum_{i=1}^N U_i(z_i(T)) - \sum_{i=1}^N C_i(y_i(1)), \tag{1}$$

$$\text{subject to:} \quad \mathbf{x}(t) \in \mathbf{R}(t), \quad \forall t, \tag{2}$$

$$\mathbf{0} \leq \mathbf{y}(t) \leq \mathbf{B}(t), \quad \forall t, \tag{3}$$

$$\mathbf{0} \leq \mathbf{z}(t) \leq \mathbf{B}(t+1), \quad \forall t, \tag{4}$$

$$\left[\sum_{j:(i,j)\in\mathcal{A}} x_{ij}(t) + z_i(t)\right] - \left[\sum_{j:(j,i)\in\mathcal{A}} x_{ji}(t) + y_i(t)\right] = 0, \quad \forall i, t, \tag{5}$$

$$\mathbf{y}(t+1) = \mathbf{z}(t), \quad t = 1, \ldots, T-1. \tag{6}$$

The objective (1) equals the net profit the network makes by selling to a hypothetical external market the data accumulated

at the end of the last epoch minus the amount it spent to buy data from that external market at the start of the first epoch. The maximization of the profit is subject to **(i)** the capacity constraints (2) for each epoch, **(ii)** the constraints (3), (4) on the storage of data at the start and end of each epoch, **(iii)** the flow conservation constraints (5) at all nodes during each epoch, and **(iv)** the requirement (6) that the contents of the buffers at nodes should not change at epoch transitions.

Regarding the costs, utilities, and external buffer sizes, various choices are possible. For example, if we set

$$C_i(y_i(1)) = 0 \quad \forall i, \quad U_i(z_i(T)) = \begin{cases} 0, & i \notin \mathcal{T}, \\ z_i(T), & i \in \mathcal{T}, \end{cases} \tag{7}$$

$$\mathbf{B}_i(1) = \begin{cases} B_S, & i \in \mathcal{S}, \\ 0, & i \notin \mathcal{S}, \end{cases} \quad \mathbf{B}_i(T+1) = \begin{cases} 0, & i \notin \mathcal{T}, \\ B_T, & i \in \mathcal{T}, \end{cases} \tag{8}$$

then our problem corresponds to finding the maximum volume of data that can be sent from a set of source nodes $\mathcal{S}$ to a set of target nodes $\mathcal{T}$, subject to the constraint that no source node can send a volume of data greater than B_S, and no target node can receive a volume of data greater than B_T.

Finally, we note that the problem may be augmented, with no complication on the subsequent analysis, by permitting the arrival and/or departure of traffic in intermediate storage nodes, and not only on the first and last set. Traffic flows leaving or arriving at the network at these storage nodes can also be associated with costs and/or utilities. As this extension is conceptually straightforward, we omit its consideration.

III. Dual Decomposition

In each of the constraints (2)-(5), there is no coupling between flows belonging to different epochs. This is not the case, however, with constraint (6). As the objective function is separable (i.e., it is the sum of functions of individual optimization variables), had constraint (6) not existed, we would have been able to solve the problem by maximizing separately over each epoch. Such problems, with weak couplings among the optimization variables, arise frequently in optimization theory [10], [11]. Here, we solve the problem using a dual decomposition method that is particularly appealing due to our problem structure.

Let $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ be the **concatenated flow vectors**, i.e., $\mathbf{x} \triangleq (\mathbf{x}(t),\ t = 1, \ldots, T)$, $\mathbf{y} \triangleq (\mathbf{y}(t),\ t = 1, \ldots, T)$, $\mathbf{z} \triangleq (\mathbf{z}(t),\ t = 1, \ldots, T)$. We define the **Lagrangian** $L(\mathbf{y}, \mathbf{z}; \boldsymbol{\lambda})$ by inserting constraint (6) to the objective function:

$$\begin{aligned} L(\mathbf{y}, \mathbf{z}; \boldsymbol{\lambda}) \quad \triangleq \quad & \sum_{i=1}^{N} U_i(z_i(T)) - \sum_{i=1}^{N} C_i(y_i(1)) \\ & + \sum_{t=1}^{T-1} \boldsymbol{\lambda}(t) \cdot [\mathbf{z}(t) - \mathbf{y}(t+1)]. \end{aligned}$$

($\mathbf{a} \cdot \mathbf{b}$ is the inner product of vectors $\mathbf{a}, \mathbf{b}$.) We call the new N-dimensional vectors $\boldsymbol{\lambda}(1), \boldsymbol{\lambda}(2), \ldots, \boldsymbol{\lambda}(T-1)$ the **price vectors**, and the $N \times (T-1)$ dimensional vector $\boldsymbol{\lambda} = (\boldsymbol{\lambda}(1), \boldsymbol{\lambda}(2), \ldots, \boldsymbol{\lambda}(T-1))$ the **concatenated price vector**. Rearranging terms, the Lagrangian can be written as:

$$L(\mathbf{y}, \mathbf{z}; \boldsymbol{\lambda}) = \sum_{t=1}^{T} \ell_t(\mathbf{y}(t), \mathbf{z}(t); \boldsymbol{\lambda}),$$

where

$$\begin{aligned} & \ell_t(\mathbf{y}(t), \mathbf{z}(t); \boldsymbol{\lambda}) \\ & \quad \triangleq \begin{cases} \boldsymbol{\lambda}(1) \cdot \mathbf{z}(1) - \sum_{i=1}^{N} C_i(y_i(1)), & t = 1, \\ \boldsymbol{\lambda}(t) \cdot \mathbf{z}(t) - \boldsymbol{\lambda}(t-1) \cdot \mathbf{y}(t), & 2 \le t \le T-1, \\ \sum_{i=1}^{N} U_i(z_i(T)) - \boldsymbol{\lambda}(T-1) \cdot \mathbf{y}(T), & t = T. \end{cases} \end{aligned}$$

Consider the following relaxation of the DTNUM Problem, that maximizes the Lagrangian:

Problem II: Relaxed DTNUM

$$\begin{aligned} \text{maximize:} \quad & L(\mathbf{y}, \mathbf{z}; \boldsymbol{\lambda}), \\ \text{subject to:} \quad & \mathbf{x}(t) \in \mathbf{R}(t), \quad \mathbf{0} \le \mathbf{y}(t) \le \mathbf{B}(t), \quad \forall t, \\ & \mathbf{0} \le \mathbf{z}(t) \le \mathbf{B}(t+1), \quad \forall t, \\ & \left[\sum_{j:(i,j)\in\mathcal{A}} x_{ij}(t) + z_i(t)\right] \\ & - \left[\sum_{j:(j,i)\in\mathcal{A}} x_{ji}(t) + y_i(t)\right] = 0, \quad \forall i, t. \end{aligned}$$

The optimization variables are the flows $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$, whereas the prices $\boldsymbol{\lambda}$ are fixed parameters. Intuitively, the network can now maximize its net profit by buying input flows and selling output flows at all epoch transitions, according to the price vectors, in addition to the start of the first epoch and the end of the last one.

As the constraints are now decoupled across time, i.e., each one involves the flows of a specific replica, in order to solve Problem II we can maximize each of the T terms of the Lagrangian independently, arriving at a set of T independent problems:

Problem III: Single Epoch Subproblem (SES) (t=1,. . . ,T)

$$\begin{aligned} \text{maximize:} \quad & \ell_t(\mathbf{y}(t), \mathbf{z}(t); \boldsymbol{\lambda}), \\ \text{subject to:} \quad & \mathbf{x}(t) \in \mathbf{R}(t), \quad \mathbf{0} \le \mathbf{y}(t) \le \mathbf{B}(t), \\ & \mathbf{0} \le \mathbf{z}(t) \le \mathbf{B}(t+1), \\ & \left[\sum_{j:(i,j)\in\mathcal{A}} x_{ij}(t) + z_i(t)\right] \\ & - \left[\sum_{j:(j,i)\in\mathcal{A}} x_{ji}(t) + y_i(t)\right] = 0, \quad \forall i. \end{aligned}$$

The optimization is over the flows $\mathbf{x}(t), \mathbf{y}(t), \mathbf{z}(t)$, with the prices $\boldsymbol{\lambda}$ being a parameter.

Consider now

$$q(\boldsymbol{\lambda}) \triangleq \max_{\mathbf{x},\mathbf{y},\mathbf{z}} L(\mathbf{y}, \mathbf{z}; \boldsymbol{\lambda}),$$

the optimal value of the relaxed Problem II. $q(\boldsymbol{\lambda})$ is a function of the prices but not the flows which are chosen to maximize the Lagrangian. By standard duality theory (see, for example,

Section 6.4.1 of [10]), there is a subgradient $\mathbf{g}(\boldsymbol{\lambda})$ of $q(\boldsymbol{\lambda})$, such that its component $g_i(t)$ corresponding to $\lambda_i(t)$ equals the violation of the respective equality constraint, i.e.,

$$g_i(t) = z_i(t) - y_i(t+1), \quad t = 1, \ldots, T-1, \ \forall i, \quad (9)$$

where the values of $y_i(t+1), z_i(t)$ come from solving the Relaxed DTNUM Problem for the given $\boldsymbol{\lambda}$.

Furthermore, assuming that the DTNUM Problem has a feasible solution with all link flow vectors strictly within their respective capacity regions (this will be satisfied in all reasonable cases), the optimal value of the original DTNUM Problem equals the optimal value of the following, unconstrained dual problem:

Problem IV: Dual DTNUM

minimize: $q(\boldsymbol{\lambda})$,

---

where the minimization is over $\boldsymbol{\lambda}$. Finally, the optimal flows, $\mathbf{x}^*(t)$, $\mathbf{y}^*(t)$, $\mathbf{z}^*(t)$ of the original DTNUM Problem coincide with the optimal flows of the Relaxed DTNUM Problem when the prices $\boldsymbol{\lambda}$ are set to the optimal value $\boldsymbol{\lambda}^*$ of the Dual DTNUM Problem. Thus, to find $\mathbf{x}^*(t)$, $\mathbf{y}^*(t)$, and $\mathbf{z}^*(t)$ it suffices to solve the Dual DTNUM Problem. Since we can calculate a subgradient at each $\boldsymbol{\lambda}$, solving the dual problem is possible with any iterative optimization method that makes use of the subgradient. One possible choice is the following algorithm:

DTNUM Algorithm

INPUT: **(i)** $\mathbf{R}(t), \ t = 1, \ldots, T$, **(ii)** $\mathbf{B}(t), \ t = 1, \ldots, T+1$, **(iii)** $C_i(\cdot)$ and $U_i(\cdot)$, $i = 1, \ldots, N$.
STEP 1 (INITIALIZATION) Initialize prices to an arbitrary initial concatenated price vector $\boldsymbol{\lambda}_0$.
STEP 2 Solve the Single Epoch Subproblems.
STEP 3 Update the prices, using their current values, the flows found in STEP 2, and equations (9).
STEP 4 IF termination condition is not satisfied, **GO TO** STEP 2, **ELSE END**.

Regarding STEP 3, various price update rules are possible. A common one, used in our simulations, is to set $\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \alpha_k \mathbf{g}(\boldsymbol{\lambda}_k)$, where k is the iteration number, $\boldsymbol{\lambda}_k$ and $\mathbf{g}(\boldsymbol{\lambda}_k)$ are the price vector and the subgradient in the k-th iteration respectively, and α_k is a step size small enough to guarantee convergence (see Section 6.3.1 of [10]). Regarding the termination condition of STEP 4, various choices are possible, e.g., terminate when the improvement to the value of the dual function over a set number of iterations is less than some $\epsilon > 0$.

IV. Numerical Evaluation

Let us now divert from theory to present a few preliminary numerical results in an illustrative wireless setting.

A. Network Model

At the start of time, N nodes are placed into a square area of side length B. All epochs have duration D. During each epoch, the nodes remain immobile, and jump to new positions on epoch transitions. To model the evolution of node positions, we sample, at the transition times, an underlying smooth mobility model, under which all nodes move continuously with a fixed common speed s and with random, uniformly chosen directions that remain constant; when a node hits the square boundaries, it gets perfectly reflected.

Regarding the communications model, nodes that are within a distance $d \leq d_0$ of each other can communicate directly through an arc a of capacity $C(a)$ equal to $C(a) = WD\log_2(1+K/d^b)$ bits per epoch, where b is a parameter of the environment (typical values are between 2 and 6) and W, K are transceiver related parameters. To address interference, we define a **maximal communication clique** $\mathcal{M}$ of nodes to be a node set such that every node in the clique can communicate directly with everyone else and, in addition, no other node can be added to the clique and maintain this property. Let $A(\mathcal{M})$ be the set of arcs that are either internal, or coming out of, or coming into a clique $\mathcal{M}$. We require the sum, over all arcs a in this set, of their rates x_a divided by the respective arc capacities $C(a)$, not to exceed unity:

$$\sum_{a \in A(\mathcal{M})} \frac{x_a}{C(a)} \leq 1.$$

In other words, the flows through these arcs are jointly constrained as if these arcs were forced to use time division. We impose this requirement on all maximal cliques that exist in the network. The resulting set of inequalities forms the capacity region. Although more refined capacity region models have been put forth [12], we adopt this model only in the context of this preliminary investigation; the precise characterization of capacity regions is beyond our scope.

The parameters chosen for the plots in this work, unless specified differently, are $N = 10$, $B = 10$, $s = 1$, $d_0 = 4$, $W = 20$, $D = 1$, $K = 100$, and $b = 4$. We place no constraints on the internal buffer sizes.

B. Results

First, we consider a simple scenario of $N = 10$ nodes. Regarding the costs, the utilities, and the buffer constraints, we adopt the model of equations (7), (8), where the set $\mathcal{T}$ consists of a single node acting as a base station, the set $\mathcal{S}$ is comprised of all other nodes, $B_S = 100$, and $B_T = 900$.

In Fig. 2 we plot the convergence to the optimal value, versus the iterations, of the dual function, the corresponding feasible profit, and the maximum feasible profit discovered in any of the past iterations. A total of $T = 5$ epochs are assumed. As is typical with the subgradient method, the convergence of the dual function to the optimal value is not monotonous and relatively slow; however, as we show next, the dual decomposition more that compensates for this slow convergence. Furthermore, observe that the maximum feasible profit discovered in any of the previous iterations converges

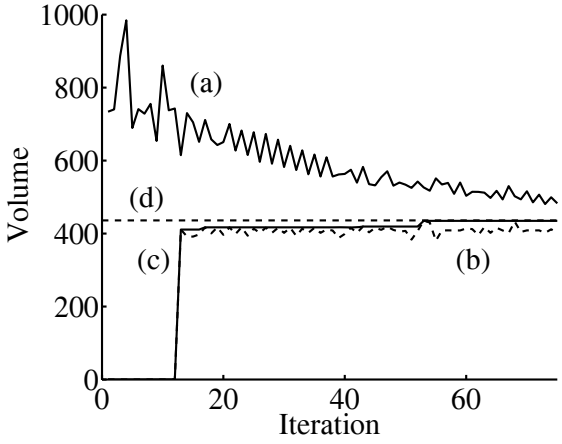

Fig. 2. Convergence of the subgradient method: **(a)** the dual function, **(b)** the feasible profit, **(c)** the maximum feasible profit discovered in all previous iterations, and **(d)** the optimal value.

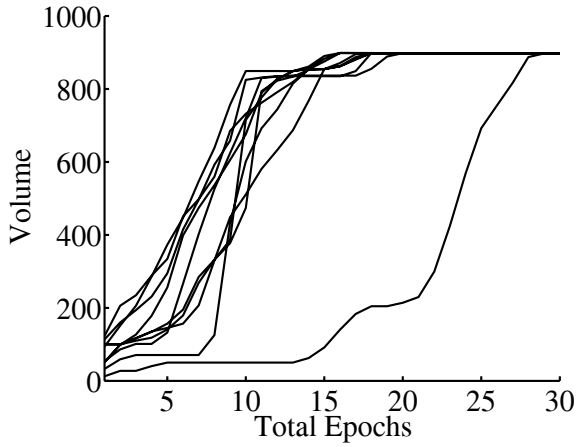

Fig. 3. Maximum volume transported versus the total number of epochs for the example network of Section IV with $N = 10$.

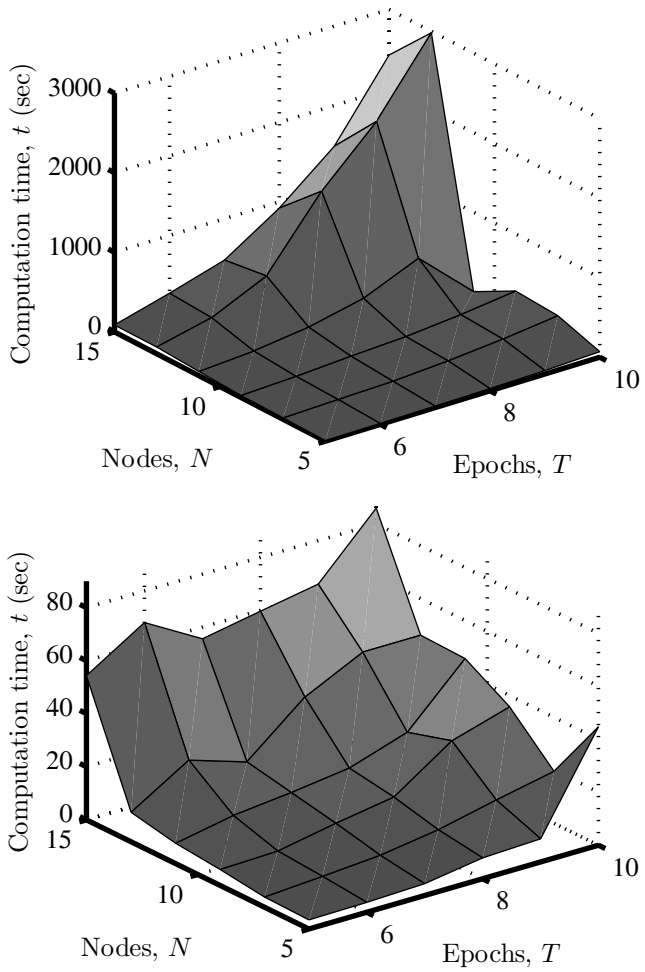

Fig. 4. Computation time t versus the number of epochs T and the number of nodes N needed for solving the DTNUM Problem directly (top) and through dual decomposition (bottom). Note the difference in the scale of the z axis.

much faster to the optimal value. These trends observed in this plot were typical of our other simulation results.

In Fig. 3 we assume the same network of $N = 10$ nodes and we plot the maximum volume transported versus the number of epochs T available, for all 10 possible choices for the base station. Observe that each of the ten curves involves solving the DTNUM Problem once for each choice of the total epochs. Note that in all cases, given sufficient time, the base station receives the maximum data volume of $9B_S = B_T = 900$. The average of these curves appears as curve (a) of Fig. 5. In the same figure we plot the average assuming that nodes move with one tenth their original speed. As expected, the received data volume increases much slower, as now data are primarily being wirelessly transmitted as opposed to physically transported.

As discussed, the primary advantage of our method is that it allows us to solve the DTNUM Problem simultaneously using multiple processors. To evaluate the gains of our method, we have solved the DTNUM Problem, using MATLAB, for a variety of combinations of the number of epochs T and the number of nodes N, and with two methods: directly, through MATLAB's generic optimization routine, and by using the DTNUM Algorithm. A total of 8 processors were used. In Fig. 4 we plot the times it took in each case for the solution to be found. Observe that even for a problem with a modest size $N = 15$, $T = 10$, the subgradient method is 30 times faster. Greater gains are expected with the use of more advanced techniques for solving the dual problem, such as the proximal cutting plane method (which we are currently investigating). Also observe that both plots are not perfectly smooth, as the amount of time needed for the routines used to converge depend to some extent on how close to the optimal flows the initial conditions are.

V. Two Causal Schemes: Greedy DTNUM and Geographic DTNUM

Observe that the optimal flow during *each* epoch depends on the state of the network in *all* epochs, including future ones.

Clearly, we would like to have a *causal* strategy for finding a feasible flow that provides a large value for the objective without looking into the future. To simplify the setting, we make here the assumptions:

$$C_i(\cdot) = 0, \ \forall i, \quad U_i(\cdot) \geq 0, \ \forall i, \quad B_i(t) \leq B_i(t+1), \ \forall i, t.$$

Therefore, it always helps for nodes to initially stock up on data, i.e., set $\mathbf{y}(1) = \mathbf{B}(1)$, even if the data end up not being transported because the capacity regions in future slots turn out to be unfavorable. Also, running out of buffer space is not possible. The assumptions are reasonable, and are actually satisfied in the case of the equations (7), (8). Given these assumptions, the following algorithm can be employed:

Greedy DTNUM Algorithm

INPUT: **(i)** Number of epochs T, **(ii)** Utility functions $U_i(\cdot)$, **(iii)** Initial buffer constraints $\mathbf{B}(1)$.
STEP 1 (INITIALIZATION) Set $\mathbf{y}(1) = \mathbf{B}(1)$ and $t = 1$.
STEP 2 Epoch t starts, $\mathbf{R}(t)$ and $\mathbf{B}(t+1)$ are given.
STEP 3 We solve the following maximization problem:

$$\begin{aligned} \text{maximize:} \quad & \textstyle\sum_{i=1}^{N} U_i(z_i(t)), \\ \text{subject to:} \quad & \mathbf{x}(t) \in \mathbf{R}(t), \quad \mathbf{z}(t) \leq \mathbf{B}(t+1), \\ & \left[\textstyle\sum_{j:(i,j)\in\mathcal{A}} x_{ij}(t) + z_i(t)\right] \\ & - \left[\textstyle\sum_{j:(j,i)\in\mathcal{A}} x_{ji}(t) + y_i(t)\right] = 0, \ \forall i, \end{aligned}$$

where the optimization variables are $\mathbf{x}(t), \mathbf{z}(t)$. ($\mathbf{y}(t)$ has been specified, either in STEP 1 or STEP 4.)
STEP 4 Set $\mathbf{y}(t+1) = \mathbf{z}(t)$, where $\mathbf{z}(t)$ was computed in STEP 3. Set $t = t+1$.
STEP 5 IF $t \leq T$, **GO TO** STEP 2, **ELSE STOP**.

Therefore, in each epoch we are greedily maximizing the sum of utilities, $\sum_{i=1}^{N} U_i(z_i(t))$, as if the epoch is the last one. Observe that costs have been assumed to be zero and so do not appear in the formulation.

In Fig. 5 we plot (curve (c)) the average data volume transported versus the number of epochs available when the Greedy DTNUM Algorithm is implemented in the example network. The difference between the two curves is expected as nodes cannot see into the future, and so cannot identify communication opportunities at later epochs.

In a wireless setting, we can improve upon the greedy strategy if we assume that nodes know the locations, speeds, and utilities of all other nodes, using in **STEP 3** the following modified objective function:

$$\sum_{i=1}^{N} \left[U_i(z_i(t)) - z_i(t) \left(\sum_{j=1}^{N} \left[k_1 d_{ij}(t) + k_2 v_{ij}(t) \right] U_j'(y_j(t)) \right) \right].$$

In the above, the summation within the parenthesis is a corrective price that, multiplied by $z_i(t)$, leads to a corrective cost subtracted from the original utility of node i and depends on that node's distance and relative velocity to all other nodes, and their utility. The cost is such, that nodes further away and/or drifting apart from suitable destinations are penalized,

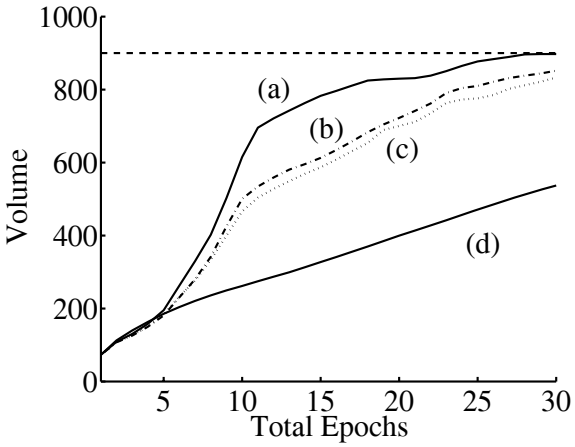

Fig. 5. Curve (a) is the average of the curves in the center. Curves (b), and (c), and (d) are the averages of the same curves but when the flows are calculated using, respectively, the Geographic DTNUM Algorithm, the Greedy DTNUM Algorithm, and the (optimal) DTNUM Algorithm when the node speed is one tenth of its value for curve (a).

by adding to their own objective function a negative linear term. In detail, $d_{ij}(t)$ is the distance between nodes i and j, $v_{ij}(t)$ is their relative speed (positive, if they are drifting apart), k_1 and k_2 are two non-negative parameters that calibrate the effects of distance and relative speed respectively, and $U_j'(\cdot)$ is the derivative of the function $U_j(\cdot)$, i.e., the rate with which the utility at node j increases as more traffic arrives there. When the derivative $U_j'(y_j(t))$ is large, sending more data to node j on top of the current volume $y_j(t)$ leads to a large increase of the net utility, and therefore node j is an attractive destination. We call the resulting algorithm the **Geographic DTNUM Algorithm**. The algorithm may be thought of as a generalized geographic routing algorithm in the case when there are multiple destinations, and data link flows are jointly constrained through the capacity regions. In Fig. 5 we plot (curve (b)) the average volume versus the number of available epochs transported under the Geographic DTNUM Algorithm in the example network. Note that the improvement is modest. Methods for achieving greater gains are currently under investigation.

VI. Multiple Commodities and Link/Storage Costs

The framework of Sections II and III can be extended to the case of multiple commodity flows. It can also be extended to include costs in the links, and costs associated with storage. Here, we provide a very brief sketch of this extension.

Assume K commodities, indexed by $k = 1, \ldots, K$. The link, input, and output flow vectors of the k-th commodity and t-th epoch are $\mathbf{x}^k(t) = (x_{ij}^k(t) : (i,j) \in \mathcal{A})$, $\mathbf{y}^k(t) = (y_i^k : i \in \mathcal{N})$, and $\mathbf{z}^k(t) = (z_i^k : i \in \mathcal{N})$. Each commodity has its own cost and utility function $C_i^k(y_i^k(1))$ and $U_i^k(z_i^k(T))$.

Assume, moreover, a cost incurred by using link (i,j) during epoch t, equal to $l_{ij}(t)$ per unit of flow and a cost incurred by storing data at node i at the end of epoch t, equal

to $s_i(t)$ per unit of flow. (We associate no cost with input flows, so as not to charge the same flow twice).

Problem V: Multicommodity/Cost DTNUM Problem

$$\begin{aligned}
\text{maximize:} \quad & \textstyle\sum_{k=1}^{K} \Big\{ \sum_{i=1}^{N} \left[U_i^k(z_i^k(T)) - C_i^k(y_i^k(1)) \right] \\
& - \textstyle\sum_{t=1}^{T} \sum_{(i,j)\in\mathcal{A}} l_{ij}(t) x_{ij}^k(t) \\
& - \textstyle\sum_{t=1}^{T} \sum_{i=1}^{N} s_i(t) z_i^k(t) \Big\}, \\
\text{subject to:} \quad & \textstyle\sum_{k=1}^{K} \mathbf{x}^k(t) \in \mathbf{R}(t), \quad \forall t, \qquad (10) \\
& \mathbf{0} \le \textstyle\sum_{k=1}^{K} \mathbf{y}^k(t) \le \mathbf{B}(t), \quad \forall t, \qquad (11) \\
& \mathbf{0} \le \textstyle\sum_{k=1}^{K} \mathbf{z}^k(t) \le \mathbf{B}(t+1), \quad \forall t, \qquad (12) \\
& \Big[\textstyle\sum_{j:(i,j)\in\mathcal{A}} x_{ij}^k(t) + z_i^k(t) \Big] \\
& - \Big[\textstyle\sum_{j:(j,i)\in\mathcal{A}} x_{ji}^k(t) + y_i^k(t) \Big] = 0, \forall i, t, K, \qquad (13) \\
& \mathbf{y}^k(t+1) = \mathbf{z}^k(t), \quad \forall t < T, \quad \forall K. \qquad (14)
\end{aligned}$$

Thus, the sums of the commodity flows must satisfy the capacity and storage constraints (10), (11), and (12), but each commodity flow must satisfy its own flow conservation constraints (13) and (14).

Again, this problem would have been separable, if it were not for constraint (14) which couples flows belonging to different epochs. If we dualize this constraint, introducing one price $\lambda_i^k(t)$ for each commodity $k = 1, \ldots, K$, node $i = 1, \ldots, N$, and epoch $t = 1, \ldots, T-1$, the objective function becomes separable, and can be optimized by solving the following single epoch subproblems:

Problem VI: Multicommodity/Cost Single Epoch Subproblem ($t = 2, \ldots, T-1$)

$$\begin{aligned}
\text{maximize:} \quad & \textstyle\sum_{k=1}^{K} \Big\{ \boldsymbol{\lambda}^k(t) \cdot \mathbf{z}^k(t) - \boldsymbol{\lambda}^k(t-1) \cdot \mathbf{y}^k(t) \\
& - \textstyle\sum_{(i,j)\in\mathcal{A}} l_{ij}(t) x_{ij}^k(t) - \sum_{i=1}^{N} s_i(t) z_i^k(t) \Big\}, \\
\text{subject to:} \quad & \textstyle\sum_{k=1}^{K} \mathbf{x}^k(t) \in \mathbf{R}(t), \\
& \mathbf{0} \le \textstyle\sum_{k=1}^{K} \mathbf{y}^k(t) \le \mathbf{B}(t), \\
& \mathbf{0} \le \textstyle\sum_{k=1}^{K} \mathbf{z}^k(t) \le \mathbf{B}(t+1), \\
& \Big[\textstyle\sum_{j:(i,j)\in\mathcal{A}} x_{ij}^k(t) + z_i^k(t) \Big] \\
& - \Big[\textstyle\sum_{j:(j,i)\in\mathcal{A}} x_{ji}^k(t) + y_i^k(t) \Big] = 0, \forall i, K.
\end{aligned}$$

(For $t = 1$ and $t = T$, the objective functions are different, and contain the cost and utility functions respectively.) Solving these problems for a fixed price vector $\boldsymbol{\lambda} = (\lambda_i^k(t) : \ i = 1, \ldots, N, \ k = 1 \ldots, K, \ t = 1, \ldots, T-1)$, gives the value of the dual function $q(\boldsymbol{\lambda})$, and a set of flows $\mathbf{x}^k(t)$, $\mathbf{y}^k(t)$, $\mathbf{z}^k(t)$, which can be used to calculate a subgradient $\mathbf{g}(\boldsymbol{\lambda})$ of the dual function using the formula

$$g_i^k(t) = z_i^k(t) - y_i^k(t+1), \quad \forall i, k, \forall t < T.$$

Therefore, we can form an iterative process that will eventually yield the optimal prices, and through them the optimal flows.

VII. Related Work

A. Dynamic Flows

In the context of Operations Research, our work is on dynamic flows and networks. In contrast to their more common, static counterparts, dynamic flows are functions of time. Ever since their introduction [13], dynamic flows have attracted a steady interest, and an impressive volume of results has been accumulated [11], [14]. The typical approach taken is to convert the dynamic problem at hand to a problem involving a static, but typically much larger in size, graph, usually called *space-time graph* or *time-expanded graph*. This was also our approach.

Typically, in dynamic networks, each arc is associated with a positive propagation delay (or latency). However, there has also been work on the special case of networks of zero propagation delays, termed *dynamic inventory networks* in [11] and *evolving graphs* in [15]. This is the assumption we make here, as, in communication networks, packetization typically leads to a propagation delay negligible with respect to the time needed for the topology to change.

The majority of the literature in dynamic flows applies combinatorial optimization techniques to solve either minimum cost or maximum flow problems, however there has been work on non-linear costs (see Section 5.6 of [11] and references therein.) To the best of our understanding, a dual decomposition technique similar to our own has not been applied, possibly because the majority of the works assume non-zero propagation delays, in which case the technique cannot be applied. However, there have been efforts to use the special structure of the space-time graphs and apply Dantzig-Wolfe decomposition; it has also been observed that in many cases dynamic networks have degeneracies, that lead to a *natural* decomposition that can be used to speed up the convergence of algorithms [16].

Next, we mention works related to our own. In the context of dynamic flows research, Ford and Fulkerson define dynamic flows in [13], and show algorithms for solving the max flow problem when arc capacities do not change with time; Hoppe and Tardos [17] develop polynomial-time algorithms for solving the quickest transshipment problem, using *chain decomposable flows*. In the context of communication networks, Ogier [18] presents a polynomial-time algorithm for maximizing the flow in a single-source, single-sink setting with piecewise linear capacities. Ferreira and his coworkers [19], [5], [15] define evolving graphs and use them to solve minimum cost problems and variants; Merugu et al. [6] use space-time graphs to compute routes for minimizing end-to-end message delivery delay. In the context of DTNs, Jain et al. introduce a modification to Dijkstra's algorithm suitable for time varying graphs with propagation delays; Hay and Giaccone [7] use a static graph of their own construction called event-driven graph to devise optimal routing strategies under various constraints; Laoutaris et al. [20] study the problem of optimizing delay tolerant bulk transfers in the Internet, also using space-time graphs.

In the above works, the aim is to maximize the flow of data transported by the network, whereas in this work we solve a utility maximization problem that contains a variety of flow maximization problems as special cases.

B. Capacity Regions and Utility Maximization

The problems of describing a capacity region and establishing if it can support a given combination of end-to-end flows has attracted significant interest in the wireless network community [21], [12], [22]. Research in the field has recently been stimulated by Network Utility Maximization (NUM) techniques extending the seminal work of Kelly et al. [23] in the wireless domain. See, for example, [24] and other works appearing on the same special issue on Nonlinear Optimization of Communication Systems.

Unfortunately, calculating if a combination of rates belongs in the capacity region or not, or trying to maximize the flow through a wireless network, are fundamentally hard problems. Indeed, even basic problems involving simple channel models can be shown to be NP-complete [25]. This is due to the inherent broadcast nature of the wireless channel, and the multiplicity of degrees of freedoms available to the nodes (in terms of who transmits, to whom, whose packets, etc.) Therefore, in the wireless DTN setting, the proposed dual decomposition is highly advantageous, as it means that, as the number of epochs becomes larger, we do not have to solve *larger* problems involving capacity regions, only *more* of them, in parallel. Furthermore, the accumulated progress in capacity region research can be immediately transfered in the wireless DTN domain, using dual decomposition.

VIII. Conclusions

We study the DTNUM Problem, a single-commodity utility maximization problem in DTNs, which we model in terms of Capacity Region Evolving Graphs (CREGs). In CREGs, rates of communication between nodes at each of T epochs are coupled, through jointly belonging in a Capacity Region associated with that epoch. Therefore, our formulation is particularly well suited to the study of Wireless DTNs. We show how to solve the problem by decomposing it into T problems, each one involving a single epoch and its associated Capacity Region. As capacity regions are very hard to describe and solve problems with [22], this decomposition is highly advantageous, as it means that we do not need to solve a single problem involving T capacity regions, but a set of T parallel problems, each involving one capacity region. As discussed, we need to solve this set of problems repeatedly, until the prices converge, however our preliminary numerical investigation reveals a significant reduction of solving times.

Acknowledgement

This work was funded by the research program 'CROWN', through the Operational Program 'Education and Lifelong Learning 2007-2013' of NSRF, which has been co-financed by EU and Greek national funds.

References

[1] K. Fall, "Delay-tolerant network architectures for challenged internets," in *Proc. ACM SIGCOMM*, Karlsruhe, Germany, Aug. 2003, pp. 27–34.
[2] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," in *Proc. ACM SIGCOMM*, Portland, OR, Aug.-Sep. 2004, pp. 145–157.
[3] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Efficient routing in intermittently connected mobile networks: the multiple-copy case," *IEEE Trans. Networking*, vol. 16, no. 1, pp. 77–90, Feb. 2008.
[4] D. Borsetti, C. Casetti, C.-F. Chiasserini, M. Fiore, and J. M. Barceló-Ordinas, "Virtual data mules for data collection in road-side sensor networks," in *Proc. ACM MobiOpp*, Pisa, Italy, Feb. 2010, pp. 32–40.
[5] A. Ferreira, "On models and algorithms for dynamic communication networks: the case for evolving graphs," in *Proc. ALGOTEL*, Méze, France, May 2002.
[6] S. Merugu, M. Ammar, and E. Zegura, "Routing in space and time in networks with predictable mobility," Georgia Institute of Technology, Tech. Rep. GIT-CC-04-07, 2004.
[7] D. Hay and P. Giaccone, "Optimal routing and scheduling for deterministic delay tolerant networks," in *Proc. IEEE WONS*, Snowbird, UT, Feb. 2009.
[8] R. C. Shah, S. Roy, S. Jain, and W. Brunette, "Data MULEs: Modeling and analysis of a three-tier system for sparse sensor networks," *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 215–233, Sep. 2003.
[9] J. Zhao and G. Cao, "VADD: Vehicle assisted data delivery in vehicular ad hoc networks," *IEEE Trans. Vehic. Technol.*, vol. 57, no. 3, pp. 1910–1922, May 2008.
[10] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA: Athena Scientific, 1999.
[11] W. B. Powell, P. Jaillet, and A. Odoni, "Stochastic and dynamic networks and routing," *Handbooks in Operations Research and Management Science*, vol. 8, pp. 141–295, 1995.
[12] S. Toumpis and A. J. Goldsmith, "Capacity regions for wireless ad hoc networks," *IEEE Trans. Wireless Commun.*, vol. 2, no. 4, pp. 736–748, July 2003.
[13] L. R. Ford, Jr. and D. R. Fulkerson, "Constructing maximal dynamic flows from static flows," *Operations Research*, vol. 6, no. 3, pp. 419–433, May-June 1958.
[14] M. Skutella, "An introduction to network flows over time," *Springer Research Trends in Combinatorial Optimization*, pp. 451–482, 2009.
[15] A. Ferreira, "Building a reference combinatorial model for MANETs," *IEEE Network*, vol. 18, no. 5, pp. 24–29, 2004.
[16] J. E. Aronson and B. D. Chen, "A forward network simplex algorithm for solving multiperiod network flow problems," *Naval Research Logistics Quarterly*, vol. 33, pp. 445–467, 1986.
[17] B. Hoppe and E. Tardos, "The quickest transshipment problem," in *Proc. Annual ACM-SIAM Symp. on Discrete Algorithms*, San Fransisco, CA, 1995, pp. 512–521.
[18] R. G. Ogier, "Minimum-delay routing in continuous-time dynamic networks with piecewise-constant capacities," *NETWORKS*, vol. 18, pp. 303–318, 1988.
[19] A. Ferreira and A. Jarry, "Complexity of minimum spanning tree in evolving graphs and the minimum-energy broadcast routing problem," in *Proc. WiOpt*, Cambridge, UK, Mar. 2004.
[20] N. Laoutaris, G. Smaragdakis, P. Rodriguez, and R. Sundaram, "Delay tolerant bulk data transfers on the internet," in *Proc. ACM Sigmetrics 2009*, Seattle, WA, June 2009, pp. 229–238.
[21] L. Tassiulas and A. Ephremides, "Stability properties of constraint queuing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Automatic Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.
[22] R. Gummadi, K. Jung, D. Shah, and R. Sreenivas, "Feasible rate allocation in wireless networks," in *Proc. IEEE INFOCOM* 2008.
[23] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control in communication networks: Shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, no. 3, pp. 237–252, Mar. 1998.
[24] B. Johansson, P. Soldati, and M. Johansson, "Mathematical decomposition techniques for distributed cross-layer optimization of data networks," *IEEE J. Select. Areas Commun.*, vol. 24, no. 8, pp. 1535–1547, Aug. 2006.
[25] A. Ephremides and T. Truong, "Scheduling broadcasts in multihop radio networks," *IEEE Trans. Commun.*, vol. 38, no. 4, pp. 456–460, Apr. 1990.