Distributed Multi-Commodity Network Flow Algorithm for Energy Optimal Routing in Wireless Sensor Networks

Jiří TRDLIČKA, Zdeněk HANZÁLEK

Czech Technical Univ. in Prague, Faculty of Electrical Engg., Dept. of Control Engg., 166 27 Prague 6, Czech Republic

trdlij1@fel.cvut.cz, hanzalek@fel.cvut.cz

Abstract. *This work proposes a distributed algorithm for energy optimal routing in a wireless sensor network. The routing problem is described as a mathematical problem by the minimum-cost multi-commodity network flow problem. Due to the separability of the problem, we use the duality theorem to derive the distributed algorithm. The algorithm computes the energy optimal routing in the network without any central node or knowledge of the whole network structure. Each node only needs to know the flow which is supposed to send or receive and the costs and capacities of the neighboring links. An evaluation of the presented algorithm on benchmarks for the energy optimal data flow routing in sensor networks with up to 100 nodes is presented.*

Keywords

Routing, In-network distributed algorithms, Multi-commodity network flow, Dual decomposition.

1. Introduction

1.1 Motivation

Our work is focused on a distributed algorithm for data flow routing through a multi-hop network. An example of the target application is a network periodically sensing some consumption variables (like gas consumption, water consumption, etc.) in large objects. Each sensing device produces a data flow of a particular volume, which is supposed to be routed through the network. The objective is to optimize the energy consumption for the data transfer (minimal possible energy consumption), while constrained by communication capacities for each communication link in the network (maximum data volume which can be transferred trough the link per a time unit).

There are many communication protocols designed for data routing in wireless sensor networks. However, to achieve energy optimal routing which complies with the communication capacities, the systems needs a central computational point with the knowledge of the actual network structure and parameters (e.g. [1], [2]). The existence of such a computational point decreases the robustness of the system against network damage and network parameters changes (e.g. link capacities, energy consumption, etc.) caused by interference or other external events. Furthermore, routing of such information (the actual network structure and parameters) has to be solved in the case of the centralized algorithm.

In this paper, we propose a distributed algorithm, which computes the energy optimal routing without the need of any central computational or data point. The algorithm supposes that each node knows only the capacity and the cost (energy consumption per sent data unit) of the communication links of the node (incoming, outgoing links) and the data which it is supposed to send and receive. The main purpose of this paper is to present the principle of new distributed routing algorithms rather than to present an application ready algorithm. We believe that the presented approach can lead to new efficient and highly adaptive routing algorithms for sensor networks.

1.2 Problem Formalization

The network topology is represented by an oriented graph where the nodes represent the devices and the oriented links represent the communication links between the devices. The communication capacity and the cost (i.e. energy consumption) are associated with the links of the graph. In the paper we assume a TDMA (Time Division Multiple Access) like protocol (e.g. GTS allocation in IEEE 802.15.4) which assigns the communication capacities and ensures collision-free communication.

The communication in the network is described by communication demands. Each communication demand is given by source nodes, sink nodes and data volume to be transferred. Therefore data transfer from several source nodes to one sink node can be described as one communication demand (multi-source). In a similar way, the model allows to describe a problem with several sink nodes (multi-sink). We formalize the problem as a minimum-cost multi-commodity network flow problem, where each commodity represents one communication demand in the network.

1.3 Related Work

Traditionally, the routing problems for data networks have often been formulated as linear or convex multi-commodity network flow routing problems e.g. [3], [4], [5] for which many efficient solution methods exist [6], [7], [8], [9]. One of the advantages of this method is that several objective functions and constraints can be put together (e.g. different types of capacity and energy consumption constraints, Real-time constraints, etc.). Using the same underlying model, we can easily combine the solution of different works focused on partial problems.

There are several works, which focus on the decomposition of network problems described by convex optimization problem. A systematic presentation of the decomposition techniques for network utility maximization (NUM) is presented in [10], [11], [12]. The authors present several mathematical approaches to structural decomposition of the NUM problems and classify them. In [13], [14], [15] the authors use dual decomposition to decompose cross-layer optimization problems into optimization of separated layers. The presented approaches lead to structural decomposition (e.g. to routing layer, capacity layer...) which is not suitable for derivation of the in-network distributed algorithm.

The decomposition of an optimal routing problem is presented e.g. in [16], [17], where the authors have focused on the node-path formulation of the routing problem and use the dual decomposition to find the distributed algorithm. The presented algorithms can be described as a negotiation between the source node and the path load. For each iteration of the algorithm the evaluation over the entire path has to be computed. This approach is suitable for problems with a small number of communication paths. However, in sensor networks like routing problems, where many possible communication paths exist, we have to find a different way to distribute the routing algorithm. Moreover, these algorithms are limited to a strictly convex objective function and fail in a case of linear objective functions.

1.4 Contribution and Outline

The main contributions of this paper are:

1. Introduction of a new distributed algorithm based on a dual decomposition of the node-link formulation of the routing problem. (The existing approaches use the node-path problem formulation, which leads to different algorithms.)

2. Introduction of a new approach to distribute a linear optimization problem by dual decomposition using the proximal-point method. (The other works using the dual decomposition of the routing problems are limited to strictly convex objective functions and fail for the linear functions.)

3. Evaluation of the presented algorithm on benchmarks for the energy optimal routing.

The paper is organized as follows: Section 2 describes the network structure, the multi-commodity network flow model for the data flow routing and the formulation of the objective function in terms of energy consumption (similar to [2]). In Section 3, which is the main part of our work, the distributed algorithm and its derivation for the energy optimal data flow routing is presented. In Section 4 the algorithm is extended into a form more suitable for node capacity constraints. An example with 100 nodes and the computational complexity experiments of the algorithm are given in Section 5. Section 6 concludes the paper and mentions the future work.

2. Multi-Commodity Model

In this section, we briefly summarize the basic terminology and specify the multi-commodity network flow model used in this paper.

2.1 Network Structure

The network is represented by an oriented graph, where for each device able to send or receive data, a node of the graph exists. The nodes are labeled as $n = 1,\ldots,N$. Directed communication links are represented as ordered pairs (n_1,n_2) of distinct nodes. The presence of a link (n_1,n_2) means that the communication, from node n_1 to node n_2, is possible. The links are labeled as $l = 1,\ldots,L$. We define the set of the links l leaving the node n as $\mathscr{O}(n)$ and the set of the links l incoming to node n as $\mathscr{I}(n)$. Each link is only in one set $\mathscr{O}(n_1)$ of some node n_1 and only in one set $\mathscr{I}(n_2)$ of some other node n_2. The network structure can be described with an incidence matrix A in the node-link form.

$$A_{n,l} = \begin{cases} 1, & l \in \mathscr{I}(n) \text{ (link } l \text{ enters node } n) \\ -1, & l \in \mathscr{O}(n) \text{ (link } l \text{ leaves node } n) \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

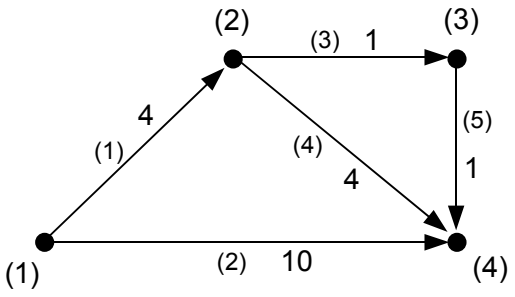

Fig. 1. Graph of the basic network.

Example: An example of a simple graph with 4 nodes and 5 links is shown in Fig. 1. The numbers in parentheses stand for the node and link indexes. The values associated to the links stand for the communication costs.

The incidence matrix A for this graph is:

$$A = \begin{pmatrix} -1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

2.2 Multi-Commodity Network Flow

We have used a multi-commodity network flow model, which is widely used in the literature of network flow routing and optimization [3], [6], [1]. In the multi-commodity network flow model, each node can send a different volume of data to any node. Each requested data transfer through the network is called the communication demand m and the set of all communication demands is labeled as $\mathscr{M}$. From the nature of the multi-commodity flow model, the data flow of each communication demand can be fragmented into more paths across the network. The model assumes that the flow is lossless and that it satisfies the flow conservation law at each node (for the given commodity, the sum of flow incoming to the node is equal to the sum of flow leaving the node).

The communication demand can be seen as a flow of a given commodity coming into the network in some nodes and leaving the network in other nodes. Each demand can come into the network in more than one node and leave the network in more nodes too (multi-source, multi-sink). Let us denote the flow volume of demand m coming into the network in node n as $s_{in,n}^{(m)} \geq 0$ and similarly the flow leaving the network in node n as $s_{out,n}^{(m)} \geq 0$. We define the vectors of the flow of demand m leaving the network as $\vec{s}_{out}^{(m)} \in R^N$ and the flow incoming into the network $\vec{s}_{in}^{(m)} \in R^N$ over all nodes. The flow coming into the network has to be equal to the flow leaving the network for all communication demands (i.e. $\sum_{n=1}^{N} s_{in,n}^{(m)} = \sum_{n=1}^{N} s_{out,n}^{(m)} \quad \forall m \in \mathscr{M}$).

Let $x_l^{(m)} \geq 0$ be the flow of demand m routed through the link l. We call $\vec{x}^{(m)} \in R^L$ the flow vector for demand m, which describes the flow of the demand in all links over the network. The flow vector and the leaving/incoming flow have to satisfy the flow conservation law for each communication demand:

$$A\vec{x}^{(m)} = \vec{s}_{out}^{(m)} - \vec{s}_{in}^{(m)} \qquad \forall m \in \mathscr{M}. \tag{2}$$

Finally, we focus on the communication capacity constraints. The vector of the total volume of the flow in the links over all communication demands is equal to $\sum_{m\in\mathscr{M}} \vec{x}^{(m)}$. There could be many different capacity constraints in the network (e.g. link capacity, node capacity, etc.). We use the link capacities in this work, which can be written as:

$$\sum_{m\in\mathscr{M}} \vec{x}^{(m)} \leq \vec{\mu} \tag{3}$$

where $\vec{\mu} \in R^L \geq \vec{0}$ is a vector of the link capacities for all links in the network.

In summary, our network flow model imposes the following group of constraints on the network flow variables $\vec{x}^{(m)}$:

$$\begin{aligned} A\,\vec{x}^{(m)} &= \vec{s}_{out}^{(m)} - \vec{s}_{in}^{(m)} && \forall m \in \mathscr{M}, \\ \sum_{m\in\mathscr{M}} \vec{x}^{(m)} &\leq \vec{\mu}, \\ \vec{x}^{(m)} &\geq \vec{0} && \forall m \in \mathscr{M}, \\ \vec{s}_{in}^{(m)} &\geq \vec{0} && \forall m \in \mathscr{M}, \\ \vec{s}_{out}^{(m)} &\geq \vec{0} && \forall m \in \mathscr{M}, \\ \vec{\mu} &\geq \vec{0}. \end{aligned} \tag{4}$$

This model describes the average behavior of the data transmission (i.e., the flow is expressed in terms of the volume of the data transmitted per a unit of time) and ignores packet-level details of transmission protocols. The link layer communication protocol (e.g. TDMA) should set the bandwidths for each demand according to the flow vectors $\vec{x}^{(m)}$. The link capacity should be defined appropriately, taking into account packet loss and retransmission, so that the flow conservation law holds with sufficient probability.

Example (continued): Let all the link capacities in our example be equal to 1. Then $\vec{\mu} = (1,1,1,1,1)^T$.

Let there be two communication demands, both with the flow volume equal to 1. The first is routed from node 1 to node 4 and the second from node 2 to node 4. Therefore, we have:
$\vec{s}_{in}^{(1)} = (1,0,0,0)^T$,
$\vec{s}_{in}^{(2)} = (0,1,0,0)^T$,
$\vec{s}_{out}^{(1)} = \vec{s}_{out}^{(2)} = (0,0,0,1)^T$.

2.3 Routing Optimization

All routings that satisfy the system of inequalities (4) are feasible solutions of the routing problem. However, the best solution in terms of the cost, needs to be found. The objective function for the optimization of the energy consumption is:

$$f_{total\ cost} = \vec{c}^T \sum_{m\in\mathscr{M}} \vec{x}^{(m)} \tag{5}$$

Where vector $\vec{c}$ is the vector of the communication cost per data unit for all links in the network. The task of the total energy minimization is to minimize the function $f_{total\ cost}$ by setting the optimal flow vector $\vec{x}$, subject to the system of inequalities (4).

Example (continued): Let the communication cost in our example be the same as in Fig. 1 (the numbers associated to the links), so $\vec{c} = (4,10,1,4,1)$.

One of the optimal solutions for this example is:
$\vec{x}^{(1)} = (1,0,0,1,0)^T$, $\vec{x}^{(2)} = (0,0,1,0,1)^T$,
which means that the first flow is routed through the nodes $(1 \to 2 \to 4)$ and the second flow is routed through the nodes $(2 \to 3 \to 4)$. The total flow is the sum of the routings over all demands $\sum_{m\in\mathscr{M}} \vec{x}^{(m)} = (1,0,1,1,1)^T$ and the communication cost is equal to 10. This energy optimal routing is shown in Fig. 2.

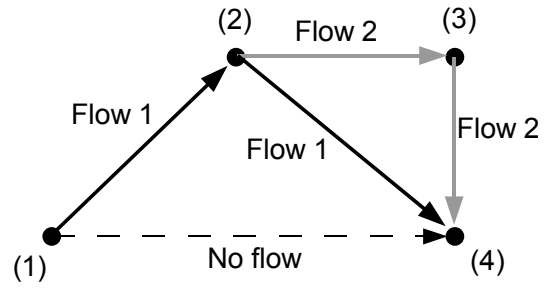

Fig. 2. An example of optimal data flow routing with capacity constraints.

3. Problem Decomposition

To decompose the routing algorithm into the network, we use a gradient optimization method to solve its dual problem. However, the linearity of the routing problem defined in Sec. 2.2 would cause oscillations in the algorithm and prevents us finding the optimal solution. We use the proximal-point method (for details see [6]) to modify the problem into a strictly convex form, which allows the usage of the gradient method. In the following text we derive the dual formulation of the routing problem and present the gradient algorithm to find the optimal solution.

3.1 Proximal-Point Method

The proximal-point method is an iterative method described as:

$$\vec{x}_{k+1} = \arg\min_{\vec{x}\in S}\left(f(\vec{x}) + \varepsilon\left(\vec{x}-\vec{x}_k\right)^T\left(\vec{x}-\vec{x}_k\right)\right) \tag{6}$$

where S is a convex set, $f(\vec{x})$ is a convex function and the $\varepsilon > 0$ is a constant. Please note, that each solution x_k of the iteration (6) is a feasible solution and that for each $k \in \mathcal{N}$ holds $f(\vec{x}_k) \geq f(\vec{x}_{k+1})$. If $k \to \infty$ then $\min_{\vec{x}\in S}(f(\vec{x}) + \varepsilon(\vec{x}-\vec{x}_k)^T(\vec{x}-\vec{x}_k)) \to \min_{\vec{x}\in S} f(\vec{x})$. The iteration (6) is called the outer iteration or the proximal-point iteration in this paper.

By applying (6) to the energy optimal routing problem and substituting $\vec{x}_k$ by $\vec{y}$, we can write the system of inequalities for one proximal-point iteration in the form:

$$\begin{aligned}
&\min_{x} \sum_{m\in\mathscr{M}} \left(\vec{c}^T\vec{x}^{(m)} + \varepsilon(\vec{x}^{(m)}-\vec{y}^{(m)})^T(\vec{x}^{(m)}-\vec{y}^{(m)})\right) \qquad (7)\\
&\textbf{subject to:}\\
&A\,\vec{x}^{(m)} = \vec{s}_{out}^{(m)} - \vec{s}_{in}^{(m)} \quad \forall m \in \mathscr{M},\\
&\sum_{m\in\mathscr{M}} \vec{x}^{(m)} \leq \vec{\mu},\\
&\vec{x}^{(m)} \geq \vec{0} \quad \forall m \in \mathscr{M},\\
&\vec{s}_{in}^{(m)} \geq \vec{0} \quad \forall m \in \mathscr{M},\\
&\vec{s}_{out}^{(m)} \geq \vec{0} \quad \forall m \in \mathscr{M},\\
&\vec{\mu} \geq \vec{0}
\end{aligned}$$

where $\vec{y}$ is the proximal-point variable with a fixed value from the proximal-point iteration.

3.2 Dual Problem

Please note that according to Slater's conditions (see e.g. [8]), strong duality holds for the problem (7) (i.e. the optimal values of the dual and the primal problem are equal).

The Lagrangian function of the system of inequalities (7) is:

$$\begin{aligned}
&L(\vec{x}^{(m)},\vec{y}^{(m)},\vec{\lambda},\vec{\theta}^{(m)}) =\\
&= \sum_{m\in\mathscr{M}} \left(\vec{c}^T\vec{x}^{(m)} + \varepsilon(\vec{x}^{(m)}-\vec{y}^{(m)})^T(\vec{x}^{(m)}-\vec{y}^{(m)})\right)\\
&+ \sum_{m\in\mathscr{M}} \vec{\theta}^{(m)^T}(A\vec{x}^{(m)} - \vec{s}_{out}^{(m)} + \vec{s}_{in}^{(m)})\\
&+ \vec{\lambda}^T\Big(\sum_{m\in\mathscr{M}} \vec{x}^{(m)} - \vec{\mu}\Big)
\end{aligned} \tag{8}$$

where $\vec{x}^{(m)} \geq \vec{0}$ is called the primal variable, $\vec{\lambda} \geq \vec{0}$ and $\vec{\theta}^{(m)}$ are called the dual variables and $\vec{y}^{(m)} \geq \vec{0}$.

The dual function W is:

$$W(\vec{y}^{(m)},\vec{\lambda},\vec{\theta}^{(m)}) = \min_{\vec{x}^{(m)}\geq\vec{0}} L(\vec{x}^{(m)},\vec{y}^{(m)},\vec{\lambda},\vec{\theta}^{(m)}). \tag{9}$$

Differentiation of the Lagrangian function (8) gives:

$$\frac{\partial L}{\partial \vec{x}^{(m)}} = \vec{c} + \vec{\lambda} + A^T\vec{\theta}^{(m)} + 2\varepsilon\vec{x}^{(m)} - 2\varepsilon\vec{y}^{(m)} \tag{10}$$

and the minimizer $\vec{x}_{min}^{(m)}$ of the dual function (9):

$$\vec{x}_{min}^{(m)} = \left[\ \vec{y}^{(m)} - \frac{1}{2\varepsilon}(\vec{c} + \vec{\lambda} + A^T\vec{\theta}^{(m)})\ \right]^+ \tag{11}$$

where $[..]^+$ denotes a positive or zero value in each component $[..]^+ = max(\vec{0},..)$. By substituting $\vec{x}^{(m)}$ in (8) by $\vec{x}_{min}^{(m)}$ from (11) we get the dual function $W(\vec{y}^{(m)},\vec{\lambda},\vec{\theta}^{(m)})$ (9).

Then the dual problem of (7) is:

$$\max_{\vec{\theta}^{(m)},\vec{\lambda}\geq 0} W(\vec{y}^{(m)},\vec{\lambda},\vec{\theta}^{(m)}) \tag{12}$$

and its gradient:

$$\frac{\partial W}{\partial \vec{\lambda}} = \sum_{m\in\mathscr{M}} \vec{x}_{min}^{(m)} - \vec{\mu}, \tag{13}$$

$$\frac{\partial W}{\partial \vec{\theta}^{(m)}} = a\,\vec{x}_{min}^{(m)} - \vec{s}_{out}^{(m)} + \vec{s}_{in}^{(m)}. \tag{14}$$

3.3 Dual Gradient Algorithm

To solve the routing problem, we put together the proximal-point method (6) and the dual problem (12). The created algorithm consists of two nested iterations. The internal iteration is the gradient iteration which solves the dual problem (12). The outer iteration is the proximal-point iteration corresponding to Equation (6). The algorithm structure is presented in Fig. 3.

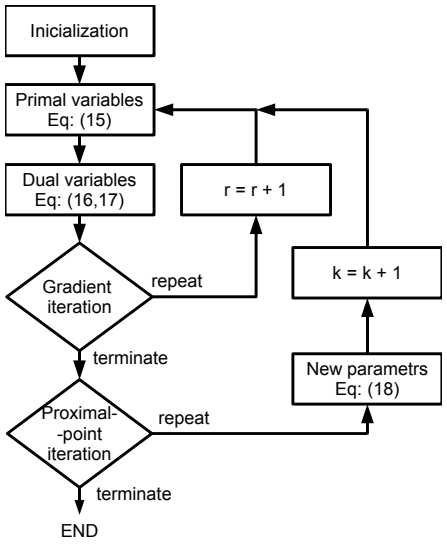

Fig. 3. Iteration algorithm.

1. Initialize the dual and the proximal-point variables:
$$r := 1, \quad k := 1, \quad \vec{\theta}^{(m)} := \vec{\theta}^{(m)}_{start},$$
$$\vec{\lambda} := \vec{\lambda}_{start}, \quad \vec{y}^{(m)} := \vec{y}^{(m)}_{start}.$$
2. Evaluate the primal variables $\vec{x}^{(m)}_{min}$:
$$\vec{x}^{(m)}_{min} := \Big[\, \vec{y}^{(m)} - \frac{1}{2\varepsilon}\big(\vec{c} + \vec{\lambda} + A^T \vec{\theta}^{(m)}\big) \,\Big]^+. \qquad (15)$$
3. Modify the dual variables according to the gradient of the dual function (13)
$$\vec{\lambda} \;:=\; \Big[\vec{\lambda} + \alpha\big(\sum_{m\in\mathscr{M}} \vec{x}^{(m)}_{min} - \vec{\mu}\big)\Big]^+, \qquad (16)$$
$$\vec{\theta}^{(m)} \;:=\; \vec{\theta}^{(m)} + \alpha\big(a\,\vec{x}^{(m)}_{min} - \vec{s}^{(m)}_{out} + \vec{s}^{(m)}_{in}\big). \qquad (17)$$
4. $r := r+1$; Go to step 2 and start a new cycle of the gradient iteration.
Repeat $R(k)$-times.
5. Start new cycle of the proximal-point iteration:
   - Set the iteration and the proximal-point variables:
$$r := 1, \quad k := k+1,$$
$$\vec{y}^{(m)} := \vec{x}^{(m)}_{min}. \qquad (18)$$
   - Go to step 2.
   Repeat the proximal-point iteration K-times.
6. Read the result routing: $\vec{x}^{(m)} = \vec{x}^{(m)}_{min}$.

Tab. 1. Dual Gradient Algorithm.

The termination of the iterations could be done through sophisticated methods based on the progress of the dual variables in combination with the global communication. However, for the verification of the concept of this algorithm it is more practical to use a constant number of proximal-point iterations based on a heuristic experiences. To the same way we set the number of the gradient iterations to decrease proportioning to the index of the proximal-point iterations.

We denote a counter of the gradient iteration as r and a counter of the proximal-point iteration as k. The constants $R(k)$ and K denote the numbers of cycles of the corresponding iterations. The constant $\alpha > 0$ is the step size of the gradient algorithm. The dual gradient algorithm is presented in Tab. 1.

In the first step of the dual gradient algorithm in Tab. 1, the variables are set to arbitrary initial values. The closer the values are to the final solution, the faster the convergence of the algorithm becomes. This property can be used in the case of minor changes of the network structure during its operation or in case of a pre-computed routing e.g. based on Dijkstra's algorithm. However, we will not investigate this problem further in this paper. During the experiments in this paper we initialize the variables to zero.

3.4 Distributed Algorithm

The presented distributed algorithm is running on each node in the network. The algorithm is synchronized by the communication between the nodes. The structure of the computation of the distributed algorithm is the same as the structure of the Dual Gradient Algorithm in Fig. 3, only the communication is added.

We use $x^{(m)}_{min,i}$, $y^{(m)}_i$, λ_i, c_i etc. to denote the i-th component of the corresponding vector.

Due to the structure of matrix A (see Sec. 2.1) we rewrite the expression (15) in order to compute the flow of the communication demand m in the link l as:
$$x^{(m)}_{min,l} = \Big[\, y^{(m)}_l - \frac{1}{2\varepsilon}\big(c_l + \lambda_l + \theta^{(m)}_{l^+} - \theta^{(m)}_{l^-}\big) \,\Big]^+ \qquad (19)$$
where the expression l^- denotes index of the start node of the link l and l^+ denotes index of the end node of the link l, i.e. $l \in \mathscr{O}(l^-)$ and $l \in \mathscr{I}(l^+)$.

Similarly we can rewrite the expressions (16, 17):
$$\lambda_l \;=\; \Big[\lambda_l + \alpha\big(\sum_{m\in\mathscr{M}} x^{(m)}_{min,l} - \mu_l\big)\Big]^+, \qquad (20)$$
$$\theta^{(m)}_n \;=\; \theta^{(m)}_n + \alpha\big(\sum_{i\in\mathscr{I}(n)} x^{(m)}_{min,i} - \sum_{i\in\mathscr{O}(n)} x^{(m)}_{min,i} - s^{(m)}_{out,n} + s^{(m)}_{in,n}\big). \qquad (21)$$

Each node n is responsible for computation of the flow volume of the links starting in the node n and for the corresponding dual variables. Therefore, node n computes $x^{(m)}_{min,l}$ for all $l \in \mathscr{O}(n)$ and all $m \in \mathscr{M}$, λ_l for all $l \in \mathscr{O}(n)$ and $\theta^{(m)}_n$ for all $m \in \mathscr{M}$.

In (19), node n computes $x^{(m)}_{min,l}$ for all links leaving node n. It is a function of the local variables y, λ, θ and the variables θ of the neighbor nodes. Similarly, the computation of λ_l and $\theta^{(m)}_n$ in (20), (21) is a function of the local variables and the variables of the neighbor nodes that are within a one hop communication distance.

The algorithm for node n is presented in Tab. 2. In step 1, the algorithm initializes the variables. Similarly as in Tab. 1, the closer the values are to the optimal solution the

less cycles of the iterations are needed. In steps 2 and 3, the node communicates the values of the proximal-point variables and the dual variables. In step 4, the node computes $x_{min,l}^{(m)}$ for the links leaving the node. In step 5, the node computes $x_{min,l}^{(m)}$ for the links entering the node from its neighbors. In step 6, the node modifies the dual variables. Steps 7 and 8 start the new iteration cycles of the algorithm.

1. Initialize the variables:
$$\begin{aligned} y_l^{(m)} &:= y_{start,l}^{(m)} && \forall m \in \mathscr{M} \quad \forall l \in \mathscr{O}(n), \\ \theta_n^{(m)} &:= \theta_{start,n}^{(m)} && \forall m \in \mathscr{M}, \\ \lambda_l &:= \lambda_{start,l} && \forall l \in \mathscr{O}(n). \end{aligned}$$
2. Send/receive the proximal-point variables to/from the neighbors.
$$\begin{aligned} \textbf{Send:} \quad & y_l^{(m)} && \forall m \in \mathscr{M}, \forall l \in \mathscr{O}(n), \\ \textbf{Receive:} \quad & y_l^{(m)} && \forall m \in \mathscr{M}, \forall l \in \mathscr{I}(n). \end{aligned}$$
3. Send/receive the dual variables to/from the neighbors.
$$\begin{aligned} \textbf{Send:} \quad & \lambda_l && \forall l \in \mathscr{O}(n), \\ & \theta_n^{(m)} && \forall m \in \mathscr{M}, \\ \textbf{Receive:} \quad & \lambda_l && \forall l \in \mathscr{I}(n), \\ & \theta_{l^-}^{(m)} && \forall m \in \mathscr{M}, \forall l \in \mathscr{I}(n). \end{aligned}$$
4. Evaluate the primal variables $x_{min,l}^{(m)}$ for $\forall l \in \mathscr{O}(n)$ and $\forall m \in \mathscr{M}$:
$$x_{min,l}^{(m)} := \Big[y_l^{(m)} - \frac{1}{2\varepsilon}\big(c_l + \lambda_l + \theta_{l^+}^{(m)} - \theta_n^{(m)}\big) \Big]^+.$$
5. Evaluate the neighbors primal variables $x_{min,l}^{(m)}$ for $\forall l \in \mathscr{I}(n)$ and $\forall m \in \mathscr{M}$:
$$x_{min,l}^{(m)} := \Big[y_l^{(m)} - \frac{1}{2\varepsilon}\big(c_l + \lambda_l + \theta_n^{(m)} - \theta_{l^-}^{(m)}\big) \Big]^+.$$
6. Modify the dual variables.
$$\begin{aligned} \lambda_l &:= \Big[\lambda_l + \alpha\big(\sum_{m\in\mathscr{M}} x_{min,l}^{(m)} - \mu_l\big)\Big]^+ && \forall l \in \mathscr{O}(n), \\ \theta_n^{(m)} &:= \theta_n^{(m)} + \\ &\quad \alpha\big(\sum_{i\in\mathscr{I}(n)} x_{min,i}^{(m)} - \sum_{i\in\mathscr{O}(n)} x_{min,i}^{(m)} - s_{out,n}^{(m)} + s_{in,n}^{(m)}\big) \\ &\quad \forall m \in \mathscr{M}. \end{aligned}$$
7. Go to step 3 and start a new cycle of the gradient iteration. Repeat $R(k)$-times.
8. Start new cycle of the proximal-point iteration:
   - Preserve the dual variables $\theta_n^{(m)}$ and λ_l for $\forall l \in \mathscr{O}(n)$ and $\forall m \in \mathscr{M}$ and set the proximal-point variables:
   $$y_l^{(m)} := x_{min,l}^{(m)} \qquad \forall l \in \mathscr{O}(n), \forall m \in \mathscr{M}.$$
   - Go to step 2. Repeat the proximal-point iteration K-times.
9. Each node n knows the routing of the outgoing flow. $x_l^{(m)} = x_{min,l}^{(m)}$ for $\forall l \in \mathscr{O}(n)$ and $\forall m \in \mathscr{M}$.

Tab. 2. Distributed Routing Algorithm executed in node n.

4. Node Communication Capacities

In some applications, especially in wireless networks, the node communication capacity (maximum data volume which can be transmitted by a node per a time unit) is more appropriate than the link communication capacity, used in this paper. The link communication capacity has been used during the algorithm derivation because of the more transparent presentation and because of the more general behavior of the algorithm (e.g. in case of the GTS slots allocation in IEEE 802.15.4 where each link has its capacity assigned). The node capacity can be easily transformed to the link capacity by the graph transformation where each node is replaced by two nodes connected by a link with the given capacity. However, in this section we present a direct transformation of the equations of the presented algorithm to implement the node capacities more efficiently.

To describe the node capacity constraints, we define the new matrix D as:

$$D_{n,l} = \begin{cases} 1, & l \in \mathscr{O}(n) \text{ (link } l \text{ leaves node } n), \\ 0, & \text{otherwise.} \end{cases} \tag{22}$$

A new form of the communication capacity constraints of Equation (3) is:

$$\sum_{m\in\mathscr{M}} D\vec{x}^{(m)} \leq \vec{\mu} \tag{23}$$

where $\vec{\mu} \in R^N$ is a vector of the node capacities for all the nodes in the network.

These changes are directly projected into the equations (4), (7), (8), (10) and (13). The changes in the dual gradient algorithm in Tab. 1 are in (15) and (16). The new form of (15) is:

$$\vec{x}_{min}^{(m)} := \Big[\vec{y}^{(m)} - \frac{1}{2\varepsilon}\big(\vec{c} + D^T\vec{\lambda} + A^T\vec{\theta}^{(m)}\big) \Big]^+ \tag{24}$$

and the new form of the (16) is:

$$\vec{\lambda} := \Big[\vec{\lambda} + \alpha\big(\sum_{m\in\mathscr{M}} D\vec{x}_{min}^{(m)} - \vec{\mu}\big)\Big]^+. \tag{25}$$

Similarly, the changes of the distributed routing algorithm executed in node n in Tab. 2 are the following:

In step 4:

$$x_{min,l}^{(m)} := \Big[y_l^{(m)} - \frac{1}{2\varepsilon}\big(c_l + \lambda_n + \theta_{l^+}^{(m)} - \theta_n^{(m)}\big) \Big]^+, \tag{26}$$

in step 5:

$$x_{min,l}^{(m)} := \Big[y_l^{(m)} - \frac{1}{2\varepsilon}\big(c_l + \lambda_{l^-} + \theta_n^{(m)} - \theta_{l^-}^{(m)}\big) \Big]^+, \tag{27}$$

in step 6:

$$\lambda_n := \Big[\lambda_n + \alpha\big(\sum_{l\in\mathscr{O}(n)}\sum_{m\in\mathscr{M}} x_{min,l}^{(m)} - \mu_n\big)\Big]^+. \tag{28}$$

While applying these changes, the algorithm uses one variable λ_n for each node instead of one variable λ_l for each link. Using a similar transformation of the equations, the algorithm can be updated to implement many different communication constraints (e.g. both oriented link capacities, node incoming and transmitting capacities...) and their combinations. Please note that when considering the link capacity constraints the matrix D is an identity matrix of size $[L \times L]$.

5. Experiments

To demonstrate the behavior and the correctness of the distributed routing algorithm, we have performed experiments in Matlab. We have focused on a data collection problem, where several nodes are supposed to send data flow to the given sink nodes (i.e. multi-commodity multi-source, mono sink problem).

The random networks for the experiments have been constructed as follows: We consider a square field of size [*size* × *size*], where the *size* is changing during the experiments. The field is divided into sub-squares of size [1 × 1]. One node is randomly placed into each sub-square and the communication distance is set to 2 (i.e. node *A* can communicate with node *B*, if and only if their Euclidean distance is less than 2). Each node, which is not on the border of the network field, has at least three communication links to its neighbors. Please notice, that our network is close to the "unit-disk network" [18]. The communication costs per transmitted data flow unit have been set as the power of the distance between the nodes.

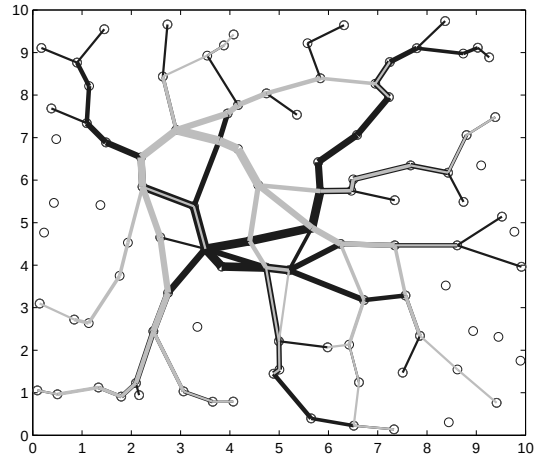

Fig. 4. Data flow routing.

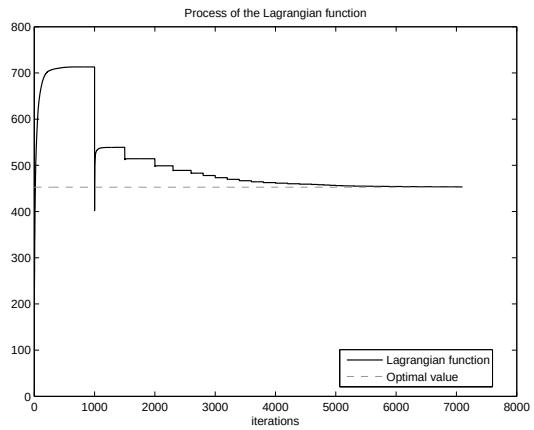

Fig. 5. Lagrangian progress.

5.1 Algorithm Presentation

To present the resulting optimal data flow routing in the network and the progress of the Lagrangian function during the computation, we have performed an experiment based on the network described above. The field *size* has been set to 10 (i.e. there are 100 nodes in the network). There are two communication demands in the network, each of them with a different sink node. Each node has a 60 % probability that it will send data of the first communication demand of volume 1 to the first sink node and a 40 % probability that it will send data of the second communication demand of volume 1 to the second sink node (i.e. the data flow coming into the network in node n of the demand 1 is $s^{(1)}_{in,n} \in \{0,1\}$ and the data flow of the demand 2 is $s^{(2)}_{in,n} \in \{0,1\}$). The link capacities have been set to $\mu_l = \frac{1}{2}\max_{m\in\mathscr{M}} \sum_{n=1}^{N} s^{(m)}_{in,n}$. The constants of the algorithm have been set as: $\alpha = 0.05$, $\varepsilon = 0.2$. The initial values $y^{(m)}_{start,l}$, $\theta^{(m)}_{start,n}$, $\lambda_{start,l}$ have been set to 0.

The optimal data flow routing is shown in Fig. 4, where the first communication demand is in black and the second communication demand in gray. The progress of the Lagrangian function (8) is presented in Fig. 5. The Lagrangian function is in black and its final value in gray. (The final value was computed separately by a centralized algorithm for evaluation purposes only.) The nesting of the gradient iteration and the proximal-point iteration can be seen in Fig. 5. The internal iteration (i.e. the gradient iteration) maximize the Lagrangian function using the dual variables $\theta^{(m)}_n$ and λ_l with constant variable $y^{(m)}_l$. The outer iteration (i.e. the proximal-point iteration) minimize the Lagrangian function using the variable $y^{(m)}_l$.

5.2 Number of Iterations

To demonstrate the statistical behavior of the algorithm, we have performed several tests in networks of different size. We have set the field *size* gradually from 3 to 10 (i.e. from 9 to 100 nodes), the number of communication demands have been set to 10 (multi-source, mono-sink) with random sinks. Each node has a 50 % probability to send the data flow for each communication demand with volume 1. The link capacities have been set to $\mu_l = \frac{1}{2}\max_{m\in\mathscr{M}} \sum_{n=1}^{N} s^{(m)}_{in,n}$. The constants of the algorithm have been set as: $\alpha = 0.1$, $\varepsilon = 0.6$. The initial values $y^{(m)}_{start,l}$, $\theta^{(m)}_{start,n}$, $\lambda_{start,l}$ have been set to 0. The computation has been repeated, on random networks, 100 times for each field *size*. The numbers of the iterations $R(k)$ and K have been set to sufficiently large values in order to ensure that each gradient iteration achieves an optimal solution for the given proximal-point variables and that the final solution is the energy optimal routing. The results have been evaluated as a maximum, average and minimum number of iterations needed to achieve a 0.01 % deviation of the Lagrangian function from

the optimal value. (the optimal value was computed separately by a centralized algorithm for evaluation purposes only).

In Fig. 6, the number of needed proximal-point iterations, in dependence on the number of nodes in the network, is presented.

In Fig. 7, the number of needed repetitions of the gradient algorithm in the first cycle of the proximal-point iteration, in dependence on the number of nodes in the network, is presented.

In Fig. 8, the number of needed repetitions of the gradient algorithm, in dependence on the index of the proximal-point method for field *size* 10, is presented. The number of the needed gradient iterations decreases rapidly during the computation. Only the first 35 proximal-point iterations are presented, because the next values are equal to 1.

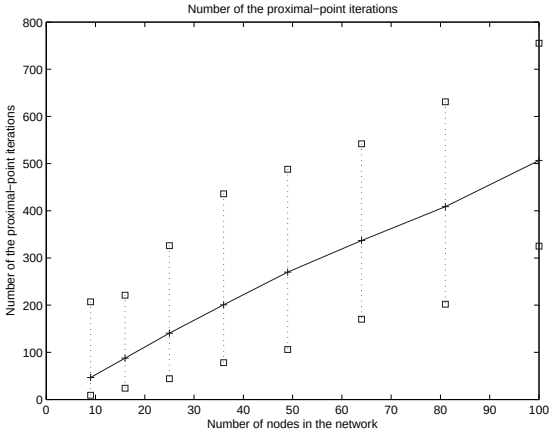

Fig. 6. Number of proximal-point iterations needed to achieve 0.01 % deviation from the optimal value.

5.3 Robustness

The presented approach requires that the gradient iteration ends with an optimal solution for a given proximal-point variables. Due to the distribution of the algorithm, the termination of the gradient iteration is problematic and in this paper we use heuristic constants for the number of repetitions. Being aware of the problems with the algorithm termination we have performed several simulations to evaluate the robustness of the approach. We have focused on the case when the gradient iterations do not reach the optimal solution for the given proximal-point variables in the given number of iterations. To ensure this behavior, we have set the number of repetitions of the gradient iterations to 50 ($R(k) = 50 \quad \forall 0 < k < K$). An example of progress of the Lagrangian function for the first 12 proximal-point iterations (i.e. 600 gradient iterations) for $R(k) = 50$ is presented in Fig. 9. All parameters for this simulation were the same as in the experiment in Sec. 5.1, except $R(k)$. In Fig. 9, it is seen that some of the first gradient iterations do not reach the optimal solution for the given proximal-point variables within the 50 repetitions. However the algorithm still converges to the final optimal solution. Moreover, it converges even faster, than in Fig. 5 since it does not spend that much time searching for the optimal solution for the given proximal-point variables.

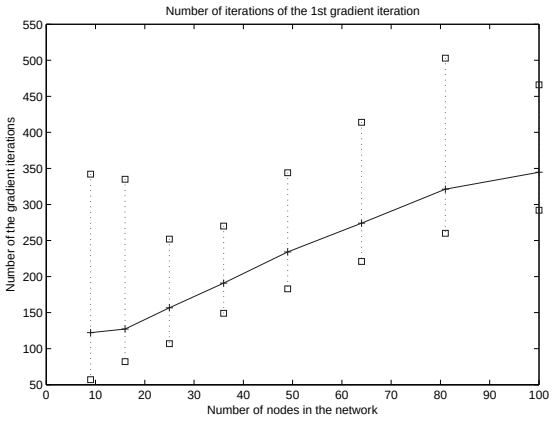

Fig. 7. Number of gradient iterations in the 1st proximal-point iteration needed to achieve 0.01 % deviation from the optimal value.

To evaluate the behavior of the algorithm in the case of incomplete gradient iterations, we have performed simulations with the same parameters as in Sec. 5.2, only the parameter $R(k)$ has been set to 50 ($R(k) = 50 \quad \forall 0 < k < K$). The experiment has been repeated 50 times for each field *size* and the results have been evaluated as a maximum, average and minimum number of iterations needed to achieve a 0.01 % deviation of the Lagrangian function from the optimal value. The number of needed proximal-point iterations, in dependence on the number of nodes in the network, is presented in Fig. 10. It is seen that the number of the needed iterations is slightly bigger than the number in the previous

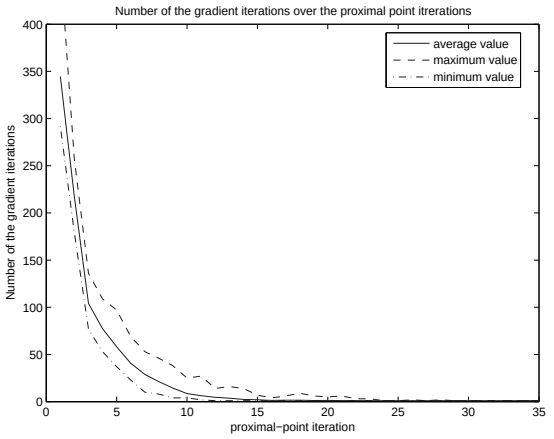

Fig. 8. Number of the gradient iterations in dependence on the index of the proximal-point iteration for the field *size* 10.

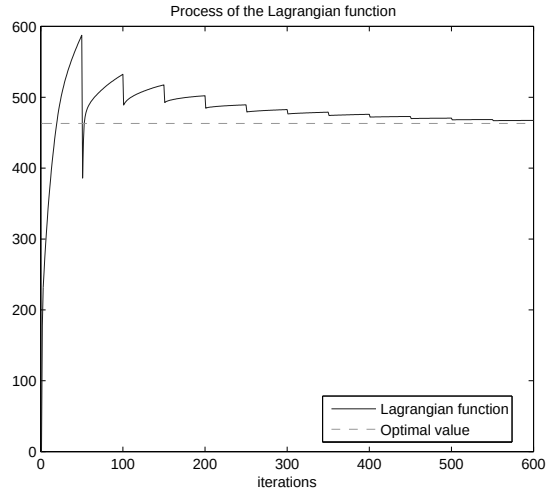

Fig. 9. Lagrangian progress for incomplete gradient iterations.

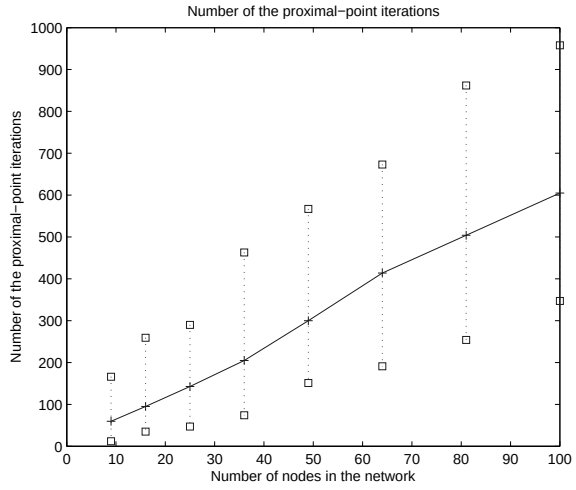

Fig. 10. Number of proximal-point iterations needed to achieve 0.01 % deviation from the optimal value in the case of an incomplete gradient iteration.

Sec. 5.2 (see Fig. 6). However the algorithm still converges to the final optimal solution.

5.4 Node Communication Capacities

To demonstrate the behavior of the algorithm in the case of the node communication capacities, described in Sec. 4, we have performed an experiment with the same parameters as in Sec. 5.2, except for the capacity constraints. The link capacities have been replaced by the node capacities $\mu_l = 3 \max_{m \in \mathcal{M}} \sum_{n=1}^{N} s_{in,n}^{(m)}$. The computation has been repeated on random networks 100 times for each field *size*.

The statistical results are presented in Figs. 11, 12 and 13 . It is seen that the progress of the number of the iterations of the algorithm for the node capacities is similar to the problem with the link capacities.

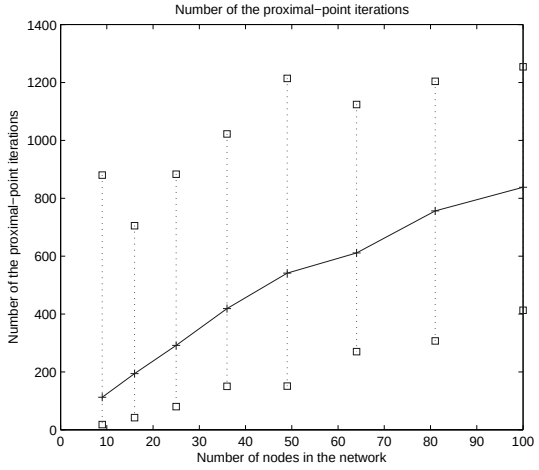

Fig. 11. Number of proximal-point iterations needed to achieve 0.01 % deviation from the optimal value for the node capacities.

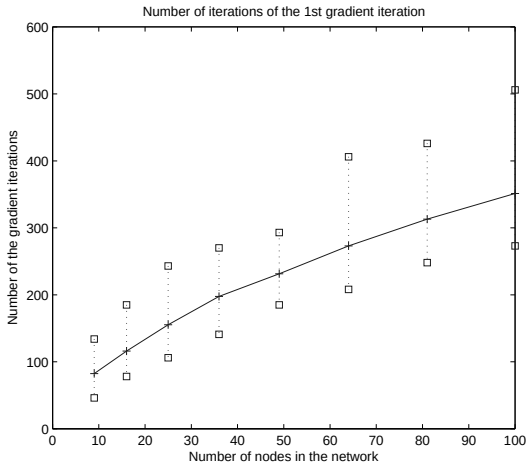

Fig. 12. Number of gradient iterations in the 1st proximal-point iteration needed to achieve 0.01 % deviation from the optimal value for the node capacities.

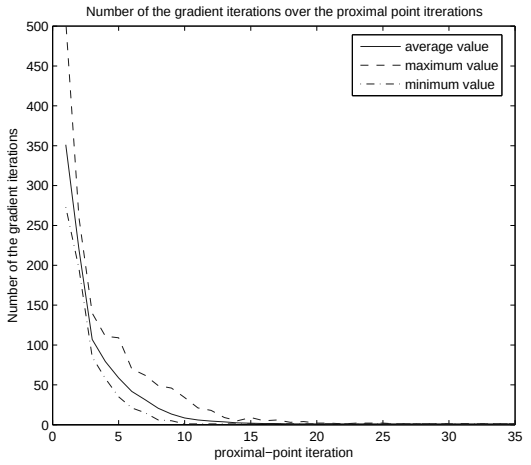

Fig. 13. Number of the gradient iterations in dependence on the index of the proximal-point iteration for the field *size* 10, for the node capacities.

6. Conclusion

In this paper we have presented a distributed algorithm for the energy optimal data flow routing in sensor networks. We have described the routing problem as a multi-commodity network flow optimization problem and used the dual decomposition method to get the distributed algorithm. The algorithm needs no central computational node, which rapidly increases the robustness of the algorithm in the case of partial network damage. The algorithm uses only peer-to-peer communication between the neighboring nodes. We believe that the principle of the algorithm and the approach used to its derivation can be used to solve many different problems in the sensor networks area, like resource sharing, network localization, object tracking, etc.

In future work we are going to improve the performance of the algorithm, using a partial knowledge about the network and extend the synchronous algorithm to an asynchronous one, which can solve the termination problem in a more practical way. Furthermore, we want to evaluate, how much the asynchronous/synchronous algorithm is able to adapt the routing subject to the network changes.

Acknowledgements

This work was supported by the Ministry of Education of the Czech Republic under the Project P103/10/0850, and Research Program MSM6840770038.

References

[1] XIAO, L., JOHANSSON, M., BOYD, S. Simultaneous routing and resource allocation via dual decomposition. *IEEE Transactions on Communications*, 2004, vol. 52, no. 7, p. 1136–1144.

[2] TRDLIČKA, J., JOHANSSON, M., HANZÁLEK, Z. Optimal flow routing in multi-hop sensor networks with real-time constraints through linear programming. In *12th IEEE International Conference on Emerging Technologies and Factory Automation ETFA07*. Patras (Greece), 2007, p. 924 – 931.

[3] BERTSEKAS, D. P., GALLAGER, R. *Data Networks*. Upper Saddle River (USA): Prentice Hall, 2004.

[4] CHIANG, M. *Geometric programming for Communication Systems - Foundations and Trends in Communications and Information Theory*. Hanover (USA): now Publishers Inc., 2005.

[5] PIÓRO, M., MEDHI, D. *Routing, Flow, and Capacity Design in Communication and Computer Networks*. San Francisco (USA): Morgan Kaufmann, 2004.

[6] BERTSEKAS, D. P. *Network Optimization: Continuous and Discrete Models*. Nashua (USA): Athena Scientific, 1998.

[7] OUOROU, A., MAHEY, P., VIAL, P. A survey of algorithms for convex multicommodity flow problems. *Management Science*, 2000, vol. 46, no. 1, p. 126 – 147.

[8] BOYD, S., VANDENBERGHE, L. *Convex Optimization*. Cambridge (United Kingdom): Cambridge University Press, 2004.

[9] JOHANSSON, M., XIAO, L. Cross-layer optimization of wireless networks using nonlinear column generation. *IEEE Transactions on Wireless Communications,* 2006, vol. 5, no. 2, p. 435 – 445.

[10] PALOMAR, D., CHIANG, M. A tutorial on decomposition methods for network utility maximization. *IEEE Journal on Selected Areas in Communications,* 2006, vol. 24, no. 8, p. 1439 – 1451.

[11] PALOMAR, D., CHIANG, M. Alternative decompositions for distributed maximization of network utility: Framework and applications. In *25th IEEE International Conference on Computer Communications INFOCOM 2006.* Barcelona (Spain), 2006, p. 1 – 13.

[12] CHIANG, M., LOW, S., CALDERBANK, A., DOYLE, J. Layering as optimization decomposition: A mathematical theory of network architectures. *Proceedings of the IEEE,* 2007, vol. 95, no. 1, p. 255 – 312.

[13] JOHANSSON, B., SOLDATI, P., JOHANSSON, M. Mathematical decomposition techniques for distributed cross-layer optimization of data networks. *IEEE Journal on Selected Areas in Communications*, 2006, vol. 24, no. 8, p. 1535 – 1547.

[14] JOHANSSON, B., JOHANSSON, M. Primal and dual approaches to distributed cross-layer optimization. In *16th IFAC World Congress.* Prague (Czech Republic), 2005.

[15] NAMA, H., CHIANG, M., MANDAYAM, N. Utility-lifetime tradeoff in self-regulating wireless sensor networks: a cross-layer design approach. In *IEEE International Conference on Communications ICC 2006.* Istanbul (Turkey), 2006, p. 3511 – 3516.

[16] TSITSIKLIS, J., BERTSEKAS, D. Distributed asynchronous optimal routing in data networks. *IEEE Transactions on Automatic Control*, 1986, vol. 31, no. 4, p. 325 – 332.

[17] LOW, S., LAPSLEY, D. Optimization flow control. I. basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, 1999, vol. 7, no. 6, p. 861 – 874.

[18] RESENDE, M. G. C., PARDALOS, P. M. *Handbook of Optimization in Telecommunications*. Berlin: Springer, 2006.

About Authors ...

Jiří TRDLIČKA was born in Jindřichův Hradec, Czech Republic in 1980 and spent his childhood in Třeboň. He received the diploma in Electrical Engineering and Cybernetics (Honors) from the Czech Technical University in Prague in 2005. Since 2005, Jiří Trdlička is working as a research worker at the Department of Control Engineering on the Faculty of Electrical Engineering of Czech Technical University, where he is working on his doctoral thesis.

Zdeněk HANZÁLEK was born in Tábor, Czechoslovakia, in 1967. He obtained the Diploma in Electrical Engineering from the Czech Technical University (CTU) in Prague in 1990. He obtained his PhD degree in Control Engineering from the CTU in Prague and PhD degree in Industrial Informatics from the Universite Paul Sabatier Toulouse. He was with LAAS - Laboratoire d'Analyse et d'Architecture des Systčmes in Toulouse (1992 to 1997) and with LAG INPG - Institut National Polytechnique de Grenoble (1998 to 2000). In 2005, he obtained Doc. degree at the Czech Technical University in Prague. He is currently a deputy head at the Department of Control Engineering at CTU and a head of the Embedded Systems Group at the Center for Applied Cybernetics.