Power Allocation with a Wireless Multi-cast Aware Routing for Virtual Network Embedding

Haitham Afifi
Computer Networks Group
Paderborn University, Germany
haitham.afifi@uni-paderborn.de

Holger Karl
Computer Networks Group
Paderborn University, Germany
hkarl@ieee.org

***Abstract*—As wireless sensor networks evolve towards comprising more powerful devices, they offer opportunities for distributing applications into the network and processing data in-network. Examples for such applications often come from the signal processing or data analysis domain. We describe such applications by overlay graphs, consisting of functional blocks with predefined interconnections that may even have feedback loops. We treat these functional blocks as virtual functions that can be easily moved (reprogrammed) among the network nodes. This property allows us to use the concept of virtual network embedding (VNE) for placement and routing. VNE has only partially been considered for wireless environments; specifically, the interaction between embedding and the wireless multicast advantage has not been fully explored. We cast our problem as mixed integer linear programming (MILP) formulation for uniform transmit power and mixed integer quadratic constraint programming (MIQCP) for power allocation and compare between both methods.**

1. Introduction

Hardware developments in Wireless Sensor Networks (WSNs) allow them to go behind data collection and do more complex data processing. Accordingly, new opportunities arose for new types of applications, which have considerable amount of data and require low delay. Examples are real-time multimedia-application scenarios [1], which have distributed sensors (microphones and cameras), collecting audio and video data, and processing the data for purposes of monitoring (e.g., classrooms), interaction (e.g., distribute microphone arrays for speaker separation), or gaming. Distributing the functions has also been exploited in low-layer networking applications as in SD-WSN [2] and MANO services [3].

In such scenarios, battery operation is not necessarily required as nodes can often be power-plugged, making energy efficiency a smaller concern than in conventional, simpler WSNs. But while tethered power supply might be ubiquitous, cable networks are not, keeping wireless communication an attractive approach.

An obvious approach for such scenarios might be to wirelessly collect all data at a central server and do all processing there. But moving some (not necessarily all) of the processing tasks from a central server to the wireless nodes has multiple advantages. First, real-time applications can require powerful and expensive processors for fast computation. Distributing some of these processing tasks to wireless nodes like Raspberry Pis or Arduinos will considerably lower the cost requirements of a central server and leverage good price/performance ratios of this class of devices. Second, and closely related, it can save wireless data rate by reducing the amount of data to communicate. Even with high-performance WLANs, this can still be a concern in dense deployments.

To support such an approach, we conceive of such applications as a graph of independent *processing blocks*, connected by data flows. A block produces and consumes data flows (possibly from data sources like microphones) and sends its output to its successor nodes. To execute such an application, two main questions need to be answered:

- *Placement*: Which node will run which block(s); a central node (e.g., acting as a gateway to external networks) with more processing power might be included in the placement decision.
- *Routing*: A data flow between blocks is mapped to which *route* between nodes.

This idea of distributing functions into a network is commonly considered in wired networks; placement and routing decisions are often based on virtual network embedding (VNE) approaches [4].

A major challenge in wireless scenarios is the interference cause by neighbouring transmissions, thus decreasing the available data rate. But this may turn into an advantage when we consider a multicast transmission from one node to many others. The ensuing embedding problem is described as **M**ulticast-**A**ware **R**outing for **V**irtual network **E**mbedding with **L**oops in **O**verlays (`MARVELO`) problem [5]. To the best of our knowledge, embedding graphs with these feature combinations has not been investigated before, especially when considering power allocation.

2. Optimisation problem

In this section, we propose a Mixed Integer Linear Problem (MILP) formulation with uniform transmit power and

978-1-5386-5553-5/19/$31.00 ©2019 IEEE

Mixed Integer Quadratic Constraint Problem (MIQCP) with power allocation. The parameters of **infrastructure graph** $(V, T, \Gamma, C, \mathrm{SINR}_{th}, N_o)$ and **overlay graph** (B, P, L, W) are defined in Table 1. we refer to [5] for further details.

Parameter	Description
v	$\in V$ node's index
c_v	$C(v)$ processing capacities of a node
$\gamma_{v,v'}$	$\Gamma(v, v')$ wireless attenuation between nodes (v, v')
t	$\in T$ time slot
$\mathrm{SINR}_{\mathrm{th}}$	minimum required SINR for a succefull transmission
N_o	noise floor
b	$\in B$ a block (or task) of signal processing
b_p	output port of a signal processing block
l	$(b_p, b') \in L$ a link between the output port p of block b to the input of block b'
w_b	$W(b)$ processing resources required by block b

TABLE 1: Parameters of the optimization problem

2.1. Decision variables

We summarize the used variables for our formulation in Table 2. We point out that the variable $s \in S$ provide a physical representation for the flow and used wireless resources (time slots), while the variable $h \in H$ virtually represents the flow, regardless the wireless resources, which facilitates the modeling of flow control in a broadcast environment.

Compared to our previous formulation [5], we add two new variables: h and λ, which are used to model the different-to-many overlays and linearizing our formulation, respectively. Moreover, we add ω only to be used in the power allocation formulation.

Variable	Description
$\theta(b, v)$	$\in \{0, 1\}$ for placing a block b on node v
$f(v, t)$	$\in \{0, 1\}$ to determine if node v is transmitting at time slot t
$s(v_1, v_2, b_p, t)$	$\in \{0, 1\}$ states that v_1 sends to v_2 the output from port p of block b at time slot t
$h(v_1, v_2, b_p)$	$\in \mathbb{N}$ to determine how many blocks needs to receive the b_p when node v_1 sends it to v_2
$\delta(t)$	$\in \{0, 1\}$ to determine if any node is transmitting in time slot t
$\lambda(v, v', t)$	$\in \mathbb{Z}^+ \bigcup 0$ a McCormick helper variable to linearize the quadratic interference between (v, v') at t
$\omega(v, t)$	$\in [0, 1]$ defines the transmit power for v at t

TABLE 2: Summary of Variables.

2.2. Constraints

We group our constraints into three main groups so that we highlight here only the new substantial ones compared to [5]. **First**, we define variable interdependency. We refer to [5] for the interdependency between s, f, and δ, while we show here the interdependency between h and s in constraints (1) and (2).

$$h(v_i, v_j, b_p) - \sum_{t \in T} s(v_i, v_j, b_p, t) \geq 0, \qquad \forall (v_i, v_j) \in V \times V, \ \forall b_p \in B_P \tag{1}$$

$$h(v_i, v_j, b_p) - \mathcal{M} \cdot \sum_{t \in T} s(v_i, v_j, b_p, t) \geq 0, \qquad \forall (v_i, v_j) \in V \times V, \ \forall b_p \in B_P \tag{2}$$

Second, we check flow constraints in the infrastructure graph. We consider in these constraints the mapping of the overlay links $l = (b_p, b')$ to a flow between nodes. Constraint (3) checks that a source node v_{src} has virtually sent the traffic of b_{src} as often as the number of successor blocks. Similarly, constraint (4) ensures that the virtually received traffic by the sink node is at least equal to the number of required traffic from the predecessors' blocks of b_{sink}. Then, we give the flow balance for any other node in (5); for a given node, the incoming traffic is equal to the outgoing traffic unless blocks are mapped on this node.

$$\sum_{v' \in V} h(v_{\mathrm{src}}, v', b_{\mathrm{src}_p}) - \sum_{(b_{\mathrm{src}_p}, b') \in L} \theta(b_{\mathrm{src}}, v_{\mathrm{src}}) = 0\,, \qquad \forall b_{\mathrm{src}_p} \in b_{\mathrm{src}_P} \tag{3}$$

$$\sum_{v' \in V} \sum_{(b_p, b_{\mathrm{sink}}) \in L} h(v', v_{\mathrm{sink}}, b_p) - \sum_{(b_p, b_{\mathrm{sink}}) \in L} \theta(b_{\mathrm{sink}}, v_{\mathrm{sink}}) \geq 0 \tag{4}$$

$$\sum_{v' \in V} h(v, v', b_p) - \sum_{(b_p, b')} \theta(b, v) = \sum_{v' \in V \setminus v} h(v', v, b_p) - \sum_{(b_p, b')} \theta(b', v), \ \forall b_p \in B_P \setminus \{b_{p_{\mathrm{sink}}}, b_{p_{\mathrm{sink}}}\}^{\forall v \in V} \tag{5}$$

Keeping track of the multicast requirements of a bock via h is the key technique that allows a linearised multicast formulation here. In other words, h virtually represents how often a block needs to be forwarded or retransmitted, and allow a node to physically duplicate the traffic (via variable s), even if the traffic is received only once.

The **Third** group concerns with wireless interference constraints and depends on the transmission power.

2.2.1. Uniform transmit power – MILP.

$$\sum_{b_p \in B_P} s(v, v', b_p, t) \cdot \mathrm{SINR}_{\mathrm{th}} \leq \frac{f(v, t)}{N_o + I(v, v')} \gamma_{v,v'}, \qquad \forall \{v, v'\} \in V, \ \forall t \in T \tag{6}$$

$$\text{where } I(v, v') = \sum_{\substack{u \in V \\ u \neq v}} f(u, t) \cdot \gamma_{u,v'}$$

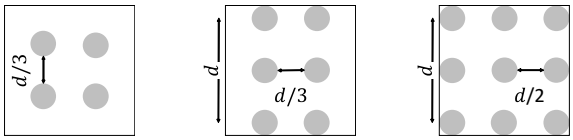
$$\frac{f(v, t) \cdot \gamma_{v,v'}}{\mathrm{SINR}_{\mathrm{th}}} \geq \lambda(v, v', t) + N_o \sum_{b_p \in B_P} s(v, v', b_p, t), \qquad \forall \{v, v'\} \in V, \ \forall t \in T \tag{7}$$

$$\lambda(v, v', t) \leq I(v, v'), \qquad \forall \{v, v'\} \in V, \ \forall t \in T \tag{8}$$

$$\lambda(v, v', t) \leq \sum_{b_p \in B_P} s(v, v', b_p, t) \cdot \gamma_{v,v'} |V|\,, \qquad \forall \{v, v'\} \in V, \ \forall t \in T \tag{9}$$

$$\lambda(v, v', t) \geq I(v, v') - |V|\,(1 - \sum_{b_p \in B_P} s(v, v' b_p, t)), \qquad \forall \{v, v'\} \in V, \ \forall t \in T \tag{10}$$

In (6), we show the formal way of controlling interference $I(v, v')$. This constraint allows transmissions from node v to v' if the SINR at v' is bigger than or equal to $\mathrm{SINR}_{\mathrm{th}}$. It will, however, yield a quadratic constraint. Thus, we reformat it in (7) to be linearly constrained, and do not need to use (6) any more. The linearisation of (6) is inspired by the McCormick method for 0/1 quadratic problems [6].



(a) 4 Nodes. (b) 6 Nodes. (c) 9 Nodes.

Figure 1: Examples of network distribution.

First, we define the effect of interference on an active edge as $\lambda(v,v',t) \leftrightarrow I(v,v') \cdot \sum_{b_p \in B_P} s(v,v',b_p,t)$. Given that the interference $I(v,v')$ is bounded between $[0,|V|]$, while $\sum_{b_p \in B_P} s(v,v',b_p,t) \in \{0,1\}$. Accordingly, we define the conditional boundaries of the integer variable $\lambda(v,v',t)$ through constraints (8) – (10).

2.2.2. Power allocation – MIQCP. We rewrite the SINR constraint to consider power allocation as in (11), but we cannot apply the McCormick method because $\omega(v,t)$ is now an integer and no longer binary as $f(v,t)$ in (6). We ensure in (12) – (13) that power allocation takes place if and only if the node is transmitting.

$$\sum_{b_p \in B_P} s(v,v',b_p,t) \cdot \mathrm{SINR}_{\mathrm{th}} \leq \frac{\omega(v,t)}{N_o + I(v,v')} \gamma_{v,v'}, \qquad \forall \{v,v'\} \in V, \ \forall t \in T$$

$$\text{where } I(v,v') = \sum_{\substack{u \in V \\ u \neq v}} \omega(u,t) \cdot \gamma_{u,v'} \tag{11}$$

$$\omega(v,t) - f(v,t) \leq 0, \qquad \forall v \in V, \ \forall t \in T \tag{12}$$

$$\mathcal{M} \cdot \omega(v,t) - f(v,t) \geq 0, \qquad \forall v \in V, \ \forall t \in T \tag{13}$$

2.3. Objective

Our **objective** is to minimize the number of used time slots within a time frame; $\min \sum_{t \in T} \delta(t)$. This reflects latency requirements of typical signal-processing or real-time applications, so that traffic is received within the first few time slots in a time frame (or it can also be used for slicing in a multi-tenant network). Other objectives are easily conceivable.

3. Evaluation and Results

We consider a seminar room, where the nodes are uniformly distributed (Figure 1). The attenuation between nodes $\gamma_{v,v'}$ is given by $\frac{1}{d^2_{v,v'}}$. The simulation is repeated for a network of 4, 6, and 8 nodes, and the nodes' capacities are changed to be proportional to the blocks' weight; $\frac{c_v}{w_b}$ ranges between 1 and 5. For each simulation, the source and sink nodes are changed. Due to page limitation, We show only the results of overlay linear topologies with blocks cascaded beside each other. The blocks' weight are fixed over all simulation, while the number of blocks ranges between 3 and 6. We evaluate the computation time required with and without power allocation and focus on the evaluation for the number of used time slots for different scenarios.

3.1. Execution Time

We set $\frac{c_v}{w_b}$ to 1 and fix the number of blocks to 5 in Figure 2a. We observe that increasing the number of nodes high impact on the computation time and considering power allocation increases the computation time significantly due to the quadratic constraint; to be in terms of hundreds of seconds (Figure 2a) and tens of seconds (Figure **??**) compared to units of seconds for uniform transmit power.

In contrast to the optimal solution in [5] and its used overlay graph, we observe that the proposed (MILP) solution here can solve the same problem for 4 and 6 nodes in 2 and 4 seconds respectively, compared to 140 and 2564 in [5]; a clear tribute to the improved problem formulation as an MILP rather than a quadratic problem.

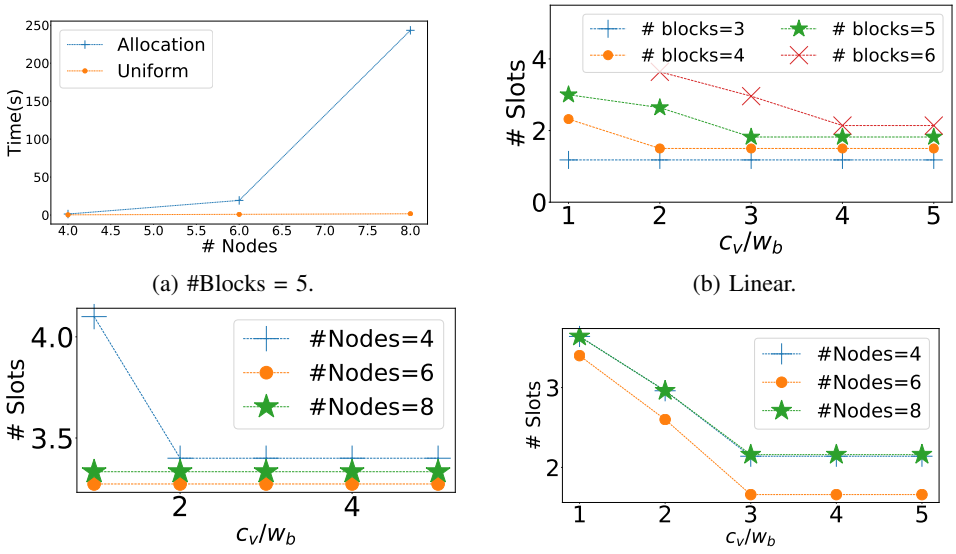
3.2. Used Time Slots

We fix the number of nodes to 5 and observe that increasing the number overlay blocks (Figure 2b) will increase the number of used time slots, due to the need for extra nodes, and consequently extra transmissions. Additionally, we observe that for all scenarios, increasing the nodes' capacity (given by $\frac{c_v}{w_b}$) decreases the number of required time slots, until it reaches a point, where the capacities are no longer a constraint. In other words, increasing the capacities of nodes allow more blocks to be placed per node, until we can place all the blocks on only two nodes; v_{src} and v_{sink}. Hence, the optimization is simplified to the routing problem, where the objective is to find a route between v_{src} and v_{sink}. The reason why we do not have results for 6 blocks at $\frac{c_p}{w_p} = 1$ is that we do not have enough resources to satisfy the capacity constraint.

Moreover, increasing the number of nodes (Figures 2c and 2d) has an impact on the number of required time slots. We observe that without power allocation (Figure 2c), increasing the number of nodes from 4 to 6, decreases the number of used time slots. Nevertheless, increasing the number of nodes from 6 to 8 increases the required time slots. A similar behaviour is observed with power allocation (Figure 2d), while this time 4 and 8 nodes require almost the same number of time slots.

Such behaviour can be explained as follow: adding more nodes will create opportunities for additional routes that possibly have lower number of time slots. Adding more and more nodes in a uniformly distributed network will, however, increase the distance between two specific nodes. This can be depicted in Figure 1, where the distance between the edge nodes increases from $\frac{d}{3}$ to d. Accordingly, it may result in extra time slots for multi-hop forwarding.

It is clear from the figures that the extra required time slots (from 6 to 8 nodes) are also related to whether power control is used or not. In case of uniform transmit power with 6 and 8 nodes, the numbers of used time slots are close to each other (8 nodes require larger number though). Meanwhile with power control, 4 and 8 nodes are having almost the same results.



(a) #Blocks = 5.

(b) Linear.

(c) Uniform Power.

(d) Power Allocation.

Figure 2: Summary of results

The results in [5] showed that increasing the number of nodes does not have a significant impact on the number of required time slots. The simulation set up here is, however, different. In this work, we distribute the nodes on a uniform grid, while in [5] the nodes were randomly uniform-distributed. Therefore, we observe such differences in the results.

4. Conclusion

In this paper we provide an optimal formulation for the wireless VNE problem, taking into account multicasts in the application and the wireless broadcast advantage. Compared to the results in [5], the new MARVELO approach can have the same results in a relatively shorter execution time. Additionally, we consider power allocation formulation and compare it to uniform transmit power.

We show through simulations that power allocation needs longer execution time but it saves in some cases 1 time slot compared to the uniform transmit power. Not only did the overlay graph's topology have a vital role in determining the required time slots, but also the number of overlay blocks and that of nodes can have an impact.

The work here gives insights about the gain when considering power allocation. That makes it suitable for benchmarking or being used in testbeds with few number of nodes. Therefore, we intend to integrate this work into our middleware [8], which uses wireless VNE to distribute tasks in a network of Raspberry Pis for realistic acoustic signal processing applications. Due to the large execution time needed to find an optimal solution for small-sized networks, further developments (e.g., a heuristic algorithm for power allocation) are still required to find an efficient solution for larger networks. Accordingly, we are looking into extending the backtrack algorithm, in our previous work, to consider power allocation as well., which will be useful for different objectives (e.g., power saving).

Acknowledgment

This work was supported by *Deutsche Forschungsgemeinschaft* (DFG) under contract no. KA2325/4-1 within the framework of the Research Unit FOR2457 "Acoustic Sensor Networks".

References

[1] Joerg Schmalenstroeer and Reinhold Haeb-Umbach. Online diarization of streaming audio-visual data for smart environments. *IEEE Journal of Selected Topics in Signal Processing*, 4(5):845–856, oct. 2010.

[2] L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo. Sdn-wise: Design, prototyping and experimentation of a stateful sdn solution for wireless sensor networks. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 513–521, April 2015.

[3] R. Riggio, A. Bradai, T. Rasheed, J. Schulz-Zander, S. Kuklinski, and T. Ahmed. Virtual network functions orchestration in wireless networks. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 108–116, Nov 2015.

[4] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach. Virtual network embedding: A survey. *IEEE Communications Surveys Tutorials*, 15(4): 1888–1906, 2013. ISSN 1553-877X.

[5] Haitham Afifi, Sébastien Auroux, and Holger Karl. MARVELO: wireless virtual network embedding for overlay graphs with loops. In *2018 IEEE Wireless Communications and Networking Conference (WCNC) (IEEE WCNC 2018)*, Barcelona, Spain, April 2018.

[6] Wajeb Gharibi and Yong Xia. A tight linearization strategy for zero-one quadratic programming problems. *CoRR*, abs/1204.4562, 2012.

[7] William E Hart, Jean-Paul Watson, and David L Woodruff. Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3(3):219–260, 2011.

[8] Afifi H, Schmalenstroeer J, Ullmann J, Haeb-Umbach R, and Karl H. Marvelo–a framework for signal processing in wireless acoustic sensor networks. *13th ITG conference on Speech Communication*, October 2018. (accepted).