Task Scheduling in Wireless Sensor Networks

Andrei Voinescu
Dan Ștefan Tudose
Nicolae Țăpuș

March 2010
Goals

- Schedule tasks in a Wireless Sensor Network

Tasks:
- Form an application
- Are repetitive
- Have inputs/outputs
- Are data centric
- May be only compatible to certain types of nodes

Schedule must also make the application as a whole to run for as long as possible
WSNs don't require a traditional scheduling problem

- Schedule isn't for distributing a large computation over many processing nodes
- The aim is to optimize the data flow and processing, from source to sink
- A more appropriate scheme would control data flow by assigning tasks to certain nodes

→ Traditional scheduling algorithms are not applicable
Approach

The schedule is in accordance with:

- Data dependencies, as specified by a DAG (Directed Acyclic Graph)
- Minimal network communication (to keep consumption low)
- Task ↔ Node compatibility

Our approach is to partition tasks from the DAG, each partition is associated with a node

Two algorithms:

- Polynomial, best solution – minimal communication between partitions
- Approximative, within twice the optimal
Energy consumption in WSNs

- Dominant component is radio communication (1 CPU instruction uses 3 orders of magnitude less energy than transmitting a bit)
  - Approaches to reduce energy consumed:
    - Distance between sensors
    - The size of data transmitted
    - Radio module uptime
  - Software can only control the latter 2
    - Radio module activity is controlled by MAC layer
    - Higher level (our scheduler) is limited to minimizing the quantity of data transmitted over the network
Problem Specifics

- Our goal → Maximize Network Lifetime

\[
\max_{k, m_k \in M} \frac{W_{m_k}}{P_{idle_{m_k}}} + \sum_{i, v_i \in T(m_k)} (P_{rcv, v_i} + P_{tr, v_i})
\]

- \( W_{m_k} \) – Total energy of node \( m_k \)
- \( P_{idle_{m_k}} \) – Idle power of node \( m_k \)
- \( P_{rcv, v_i} \) – Power consumed by task \( v_i \) while receiving
- \( P_{tr, v_i} \) – Power consumed by task \( v_i \) while transmitting

\( T \subseteq m_k \) – The set of tasks allocated to node \( m_k \)
Problem Specifics (II)

- The solution → find a partitioning of the set of tasks over existing nodes that minimizes communication taking into account the other constraints.
Constraints

- Minimal communication between nodes
- Task compatibility with nodes
- Multiplicity of tasks in application:
  - On only one node
  - On all compatible nodes
  - On as many nodes as necessary
Partitioning - (s,t) Cut

- Partitioning (S,T) of a set of nodes in a graph G, such that:
  \[ G = \{V, E\} \subseteq S \cup T, V = S \cup T, S \cap T = \emptyset \]
  \[ E' = \{ (x, y) \in E \mid x \in S, y \in T \}, \quad \sum_{e \in E'} w(e) \text{ is minimal} \]

- Can be extended to multiple-source multiple-destination

- Equivalent to maxflux problem
Based on (s,t) cuts, with s,t as sets
- Generates all set possibilities
  - s with $k-1$ elements if $k$ is odd
  - s with $k-2$ otherwise
  - t with $k-1$ elements
- An (s,t) is computed for each
- The set that contains s is kept
- The set that contains t is split into $k-1$ (min $k-1$)

- $O(n^{k^2}) \rightarrow$ polynomial, but large
Adapted min k-cut Algorithm

- Adaptation of Min k-cut
- Tasks with multiplicity must appear on all compatible nodes
- The set for the current component (first of the k in this recursive call) can only contain compatible tasks
- Recursive call with k-1

```java
function AKCut(V, k, m_i)
if k is even then
    k' = k - 2
else
    k' = k - 1
end if
MT ← tasks that have multiplicity
V' ← V - MT
S ← the set of subsets of k' elements from V'
T → the set of subsets of k - 1 elements from V' \ MT
Find \( s \in S, t \in T \) such that \( W(\text{cut}(s, t)) = \min \)
/* cut(s, t) splits V into s' and t' */
/* Find the minimal cut(s, t) with maximal source set */
\( T(m_i) = \{ s' \} \cup \{ v_j | v_j \in MT, N A(v_j, m_i) = 1 \} \)
return \( T(m_i) \cup \text{AKCut}(V-s', k-1, m_{i+1}) \)
```
Approximation algorithm

- **Gomory-Hu Trees:**
  - Each edge has the weight of an (s,t) cut
  - Elimination of the smallest k-1 edges
  - At least k components are obtained
  - (a,b) cut is the same as the smallest edge on chain between a and b
## Related Work

### EcoMaps

<table>
<thead>
<tr>
<th>Similarities</th>
<th>Differences</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Application is divided into tasks</td>
<td>• Tasks are considered to be a part of a larger computation with a precise end, i.e. FFT</td>
</tr>
<tr>
<td>• Dependencies between tasks are marked by an Acyclic Directed Dependency Graph</td>
<td>• Time constraint dominates energy consumption</td>
</tr>
<tr>
<td>• One of the constraints is energy</td>
<td>• Simplified Communication model, with many single-hop clusters</td>
</tr>
<tr>
<td></td>
<td>• EcoMapS solves a more traditional scheduling problem</td>
</tr>
</tbody>
</table>
Software Platform: Contiki

- Built for wireless sensors
- Collaborative processes
  - Based on coroutines
- uIPv6 communications stack (6lowPan)
- Communication abstractization → protosockets
- Hardware abstraction layer
Conclusions

- Dependency graph is a good model for data dependencies between tasks
- Min k-cut offers minimal communication, but is not scalable
- Approximation algorithm good for practical purposes