A Cross-Layered Communication Protocol for Load Balancing in Large Scale Multi-sink Wireless Sensor Networks

Ayşegül Tüysüz Erman, Thijs Mutter, Lodewijk van Hoesel, and Paul Havinga

Department of Electrical Engineering, Computer Science and Mathematics,
University of Twente
Postbus 217, NL-7500 AE Enschede, The Netherlands
{A.Tuysuz, T.F.Mutter, L.F.W.Vanhoesel, P.J.M.Havinga}@utwente.nl

Abstract. One of the fundamental operations in sensor networks is convergecast which refers to the communication pattern in which data is collected from a set of sensor nodes and forwarded to a common end-point gateway, namely sink node, in the network. In case of multiple sinks within the network, the total load of the network has to be balanced among these sinks to minimize the problem of packet loss in the convergecast process in wireless sensor networks (WSNs) due to congestion and collisions near the sinks. In this paper, we present a novel cross-layered communication protocol for efficient data dissemination in multi-sink WSNs which is under consideration of SENSEI project. It basically combines network wide load balancing, clustering techniques and local routing optimizations with SENSEI architecture which make it efficient on both global and local level. The performance evaluation of the proposed technique shows how our routing protocol can balance the network load without additional control packets for routing tree maintenance.

Keywords: Cross-layered communication architecture, load balancing, multiple sinks, constraint-based routing, WSN.

1 Introduction

Advances in sensor technology and wireless communications certainly open the way to a wide range of applications for environmental monitoring, traffic control, building management, object tracking, etc. In wireless sensor networks (WSN), each sensor individually senses the environment, but collaboratively achieves complex information gathering and dissemination tasks. Different applications running on WSNs have different requirements for data collection/disseminations process. For example, a networking metric can be short message delay for time-critical applications; on the other hand, it can also be minimum energy usage for environmental (i.e. agricultural field, underwater, etc) monitoring. Therefore, it is mandatory to design communication protocols which are aware of application demands and can adopt themselves according to the application-specific features of WSNs. It is the aim of the EU-funded project SENSEI [1] to bring forward a highly scalable architecture which enables integration of application requirements and real-world resources (i.e. sensor, sinks, and actuators, etc.) with in-network processing.

Typically wireless sensor network follows the communication pattern of convergecast, where sensors relay streams of data either periodically or based on events to a common sink node which is a network-layer gateway having the functionality of communicating (routing) between WSN and the management system. In case of large deployment areas, the sink can not be reached by all the sensors in the system. Usage of multiple sinks (multi-sink) appears as an efficient solution for large scale networks. We explore the certain benefits of having multiple sinks in the network as follows:

Energy efficiency: In large scale WSNs, long routing path lengths from sensors located at the network borders to the sink are observed. Adding extra sinks to the network decrease the average path length between a sensor and the sink due to shorter geographic distance between them. Therefore, the number of hops that a packet has to travel to reach a sink gets smaller. Since each traveled hop means the data packet consumes some energy at the visiting node, travelling fewer hops results in consuming less energy.

Avoiding congestion near a sink: Using multiple sinks can also relieve the traffic congestion problem associated with a single-sink system as illustrated in Fig. 1.

Avoiding single point of failure: A single-sink WSN is not robust against failure of the sink or the sensor around the sink. Multi-sink networks are therefore more resilient to node failures. However, deploying more sink nodes does not solve the problem directly and evenly. It is essential to distribute network load among sinks and choose an optimal route(s) between sensors and the corresponding sink.

This paper proposes a partition-based network-load balancing (P-NLB) protocol for SENSEI architecture that takes the shortest-path routing as the based point and uses application specific load sensitive metrics for routing from sensors to sinks. Since the load balancing optimization problem in a multi-hop network is NP-hard [2, 3], we propose a heuristic algorithm. In this paper, we have the following contributions:

- **Cross-layered communication architecture for WSNs:** We present the general architecture of SENSEI and show how it enables resource discovery (i.e. sensor/sink discovery) and integration of application requirements with a cross-layer approach for load balancing and routing.
- **Performance evaluation of multi-sink WSN:** We show the benefits of having multiple sinks on some network performance metrics by extensive simulations.
- **Load balancing among sinks:** It is applied between partitions to achieve load balancing among sinks globally.

- **Metric-based routing within each cluster:** After load balancing, P-NLB uses different routing metrics provided by SENSEI Application Subsystem to establish route between sensors and sinks.

The remainder of this paper is organized as follows. Section 2 discusses the related work. In Section 3, we explain the WSN model of SENSEI. Section 4 describes our protocol P-NLB. Performance evaluation is presented in Section 5. We conclude the paper and present future work in Section 6.

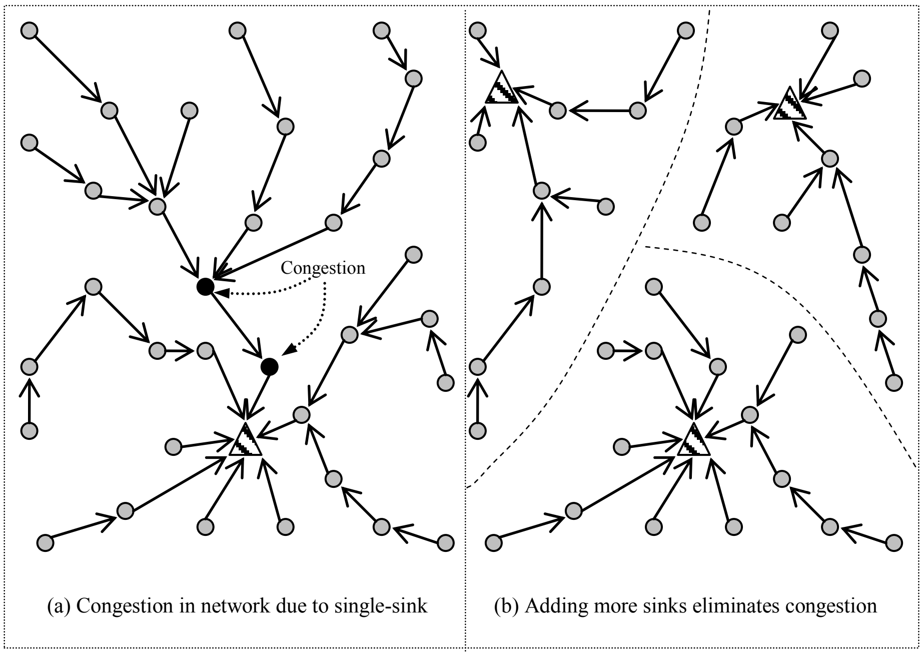

Congestion
(a) Congestion in network due to single-sink
(b) Adding more sinks eliminates congestion

Fig. 1. Comparison of single- and multi-sink sensor networks

2 Related Work

Table 1 gives an overview of some related work and their properties related to our research problem definition.

Table 1. Overview of related works.

Protocol	Protocol Type	Objective	Multi-sink
ART [4]	Routing	Parent Selection	No
LBC [5]	Clustering	Load Balancing	Yes
Distributed LBR [6]	Routing	Load Balancing	No
Node-Centric LB [7]	Routing	Load Balancing	No
e3D [8]	Routing	Load Balancing	No
GLBCA [9]	Clustering	Load Balancing	Yes

A common routing technique for WSNs is building multiple spanning trees in which sensors are vertices and forwarding vectors are edges. Each spanning tree has one sink which is the root of that tree. Such a tree is also called cluster or partition in the literature. Spanning trees for routing are used by most of the existing works [4, 6, 7]. Each sensor other than sink has a pointer to its parent which is one of its neighbors. The procedure of deciding which of the neighboring sensor nodes will be the current sensor's parent is called *parent selection*.

In ART [4], a cost function called *Q-value* for each node is defined according to routing specification of a message. This function indicates the minimum cost-to-go from the current sensor to the sink with a given routing objective. Furthermore, a sensor also stores its neighbors' Q-values, *NQ-values*, which are updated when packets are received from neighbors. A spanning tree is constructed in the initialization phase of ART. Each sensor selects its parent which is the neighbor with the smallest *NQ-value*. In ART, the parent selection is based on different routing specifications such as energy-awareness and congestion-awareness. ART does not make use of multiple sinks and does not deal with load balancing; it only tries to use different routing paths towards sink according to different routing requirements. Therefore, its routing concept, also used in [10], seems effective. We will use a similar approach to route data from sensors to sinks in each partition.

For network load balancing, various techniques [5-9] have been proposed in the literature. In LBC [5] and GLBCA [9], the distribution of the load is controlled by clustering algorithms. Each cluster in the network has a cluster head which gathers data from sensors within the cluster. In LBC and GLBCA, the network contains multiple sinks, each of which is also a cluster head. LBC uses energy reserves and locations of sensors to balance load among sinks. In [6], the goal is to distribute the energy consumption in the network. It is achieved by forwarding data to sensors which have a high energy level. In [7], the authors look at the structure of the routing paths from sensors to the sink and use an offline method for balancing the load across different branches of the routing trees. Both [6] and [7] have only one sink in the network and try to balance routing trees rooted at this sink. The e3D [8] uses the distance between each sensor and the sink as a routing metric to forward data to sink. In its diffusion based approach, a sensor can order other sensors to stop using it as a relay node, for example, if its energy level is below a certain threshold.

All of the discussed protocols have certain drawbacks which make them unsuitable for load balancing in large scale multi-sink WSNs. The protocols in [5], [7], [9] are centralized which makes them not scalable for large networks. Also, they are not flexible in topology and network condition changes. Some of them [5], [8], [9] assume the availability of location information of the sensors needing a GPS device or a localization algorithm on the sensors. As shown in Table 1, most of them are designed for single-sink networks and can not be efficient in multi-sink systems. Finally, we emphasize that none of them handles both global load balancing among sinks and metric-based convergecast in global cross-layer communication architecture.

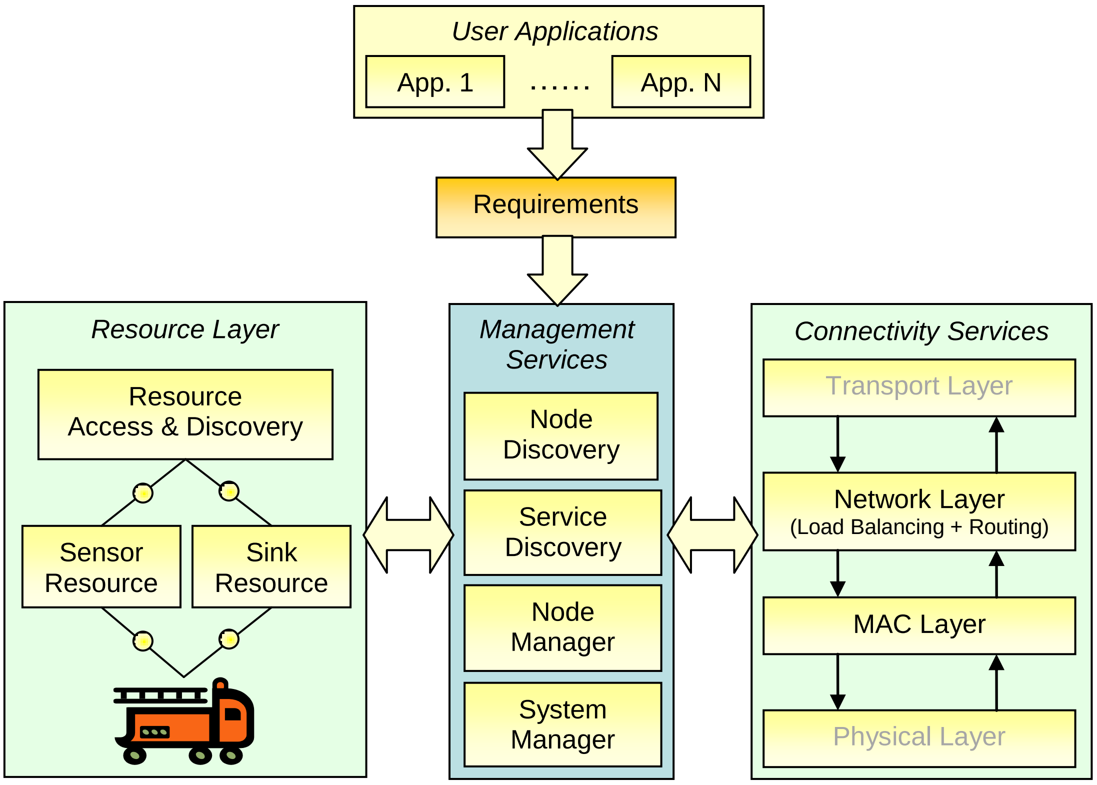


Fig. 2. The Cross-layered communication architecture

3 The Cross-layered Communication Architecture and WSN Model

Following the SENSEI project general approach, we make a first step forward in this paper and describe application-oriented cross-layer communication protocol for WSNs which have multiple sinks. In this example, the ultimate goal is to distribute the network load among multiple sinks, thus discovering resources (end-point nodes – sensors and end-point gateways – sinks), enabling the system to detect network topology (number of clusters, number of sensors in each cluster, etc.), and adjustment of application-aware routing paths between sensors and sink in each cluster. The resulting framework forms the basic services as shown in Fig. 2, which can exploit the functionality of the SENSEI system.

This framework provides management functions to reprogram the protocol stack running within the WSN nodes (i.e. sensors and sinks) in part or entirely. This enables the modification of connectivity functions and the application-aware networking which governs the gathering of data. The management features can be defined to support the variety of requirements derived from application scenarios. For example, a "Crisis Management" scenario requires a tightly managed system and very short latency in the networking. Therefore, these requirements should be mapped into management and connectivity services (i.e. networking requirements of application are mapped into routing metrics in networking layer).

Discovery and management of WSN internals are also important components of the system and they are required to achieve topology awareness and control. First, real world resources (e.g. a fire truck in a "Crisis Management" scenario) are mapped into WSN resources (i.e. sensor and/or sink on the fire truck) in the Resource Layer. After "Node & Service Discovery" functionalities get information about WSN resources, "Node & System Managers" process this information. System Manager detects the general topology of the network with coordination with connectivity functionalities. It then updates connectivity functionalities according to network topology. For example, if an unbalanced network (i.e. each sink has different load) is detected, load balancing functionality is triggered in networking layer.

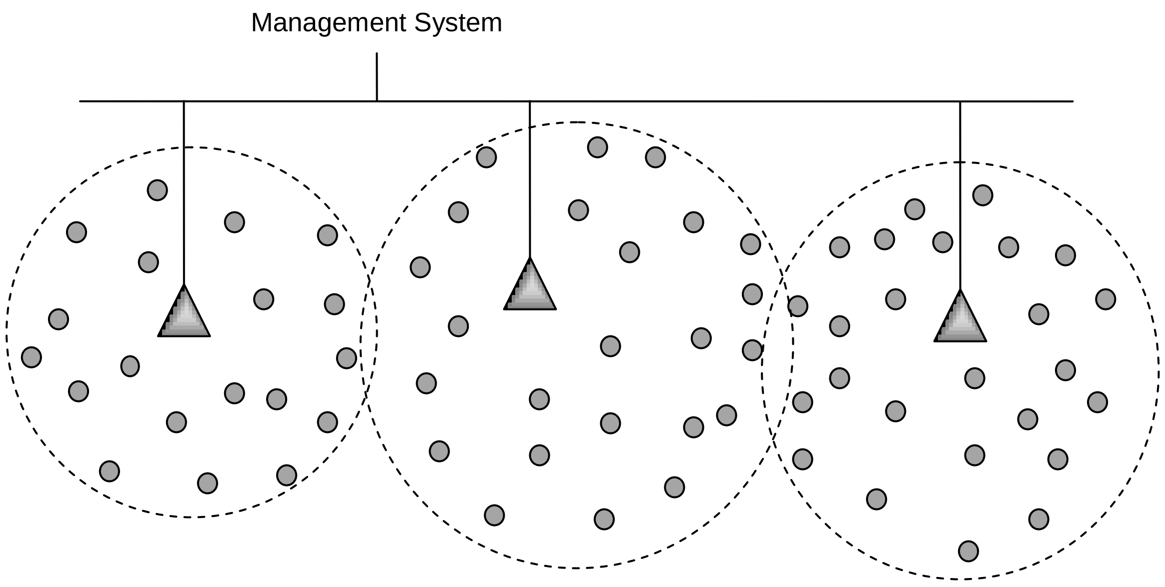


Fig. 3. Multi-sink WSN topology in SENSEI system

In this paper, we consider the WSN model shown in Fig. 3. The WSN consists of multiple sinks and many sensor nodes which are *stationary* and form a multi-hop network. Sinks can *directly communicate* with each other using a high-speed communication channel provided by the SENSEI architecture. Sinks are equal from the information point of view; it does not matter to which sink a data packet is sent. We assume that after reception of the packets, all sinks forward them to the Management system. Also, it is possible to exchange *cross-layer information* between data link layer (MAC) and network layer as shown in Fig. 2.

LMAC [11] is used as underlying MAC protocol, providing cross layer information for our load-balancing and routing algorithm. LMAC is a TDMA-based lightweight medium access control protocol design especially for WSNs. In LMAC, time is divided into frames, each of which is further divided into a fixed number of time slots. Every node chooses its own slot using a distributed algorithm which uses only locally available information. A node is allowed to pick any slot as long as it is not owned by other node within its two-hop neighborhood. This mechanism makes it possible for two nodes which are two hops away each other to transmit at the same time. Although LMAC is used as MAC, our approach is independent from LMAC, since any other MAC protocol which provides the same cross-layer information – as described below – can be used with our protocol.

Here, we assume that at the beginning of start up, the WSN is organized autonomously. In this setup phase, all sensors in the network initialize themselves by LMAC protocol. LMAC provides two useful pieces of information: (i) Neighborhood information, and (ii) Distance to sink(s), for each sensor. After this phase, each sink also has information about initial number of sensors which are connected to it and the other sinks. This information is very useful to determine the actual need of balancing the network. The steps of setup phase are shown in Fig. 4.

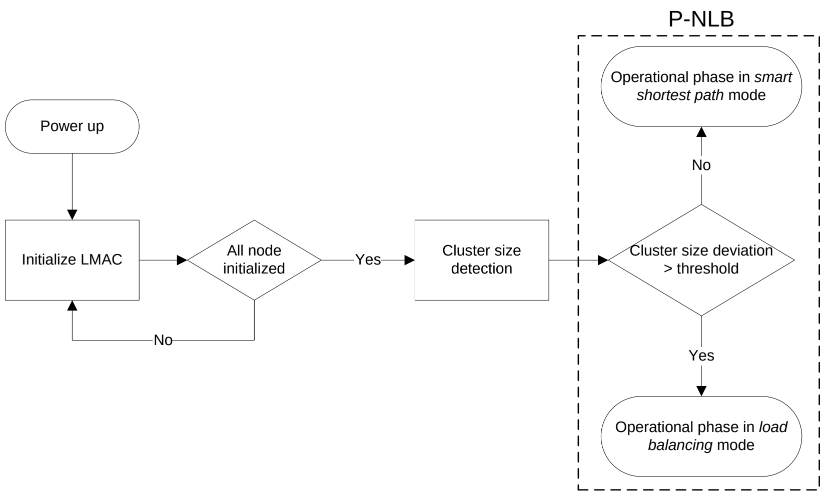

P-NLB
Power up
Initialize LMAC
All node initialized
Yes
No
Cluster size detection
Cluster size deviation > threshold
No
Yes
Operational phase in *smart shortest path* mode
Operational phase in *load balancing* mode

Fig. 4. State diagram of setup phase

4 Partition-based Network Load Balanced Routing (P-NLB)

P-NLB is a novel routing protocol that uses a two-tier approach that combines routing on *local level* with a load balancing technique on a network wide *global level*. On the local level sensors use the cross-layered approach to exchange information with their one-hop neighbors and get a view of their *local* neighborhood.

In WSNs, the most basic routing approach is the shortest path routing (SPR) paradigm to send data packets to the sinks. However, in multi-sink networks, SPR does not guarantee that the resulting spanning tree is load balanced. SPR, which minimizes the number of hops a packet travels, leads to the forming of spanning trees containing different amount of sensors, because selecting the shortest path does not account for the effect of load aggregation on upstream links. Therefore, by assuming uniformly generated load per node, SPR creates spanning trees with different loads in the network.

Although the base point of P-NLB is SPR, it also uses other metrics to construct spanning trees rooted at sinks. P-NLB uses an approach which:

- Is distributed. Each sensor decides for itself to which sink it will route its data. Therefore, each sensor knows which cluster it belongs to.
- Fully utilizes the existing of multiple sinks in the network. It features not only inter-cluster load balancing, but also intra-cluster metric-based routing.
- Does not need explicit network maintenance. Routing tree is very flexible and adapts itself easily according to changes in the network.
- Scales very well for large sensor networks since it has very low communication overhead.
- Needs no geographical location information.

4.1 Adaptive Routing Mode with Cluster Size Distribution Detection

Detecting the global network structure is important to decide whether the load balancing is needed or not in the network. For the networks, which have equal size of clusters, SPR without load balancing is sufficient to forward data to the sinks. Inter-cluster load balancing is only necessary in networks which have the typical asymmetric shapes with large and smaller clusters. Therefore, to choose the right operational mode of P-NLB, detection of the network structure is essential.

As part of the LMAC setup phase (see Fig. 4), each sensor node uses a simple one-time broadcasting technique to detect the closest sink and reports its presence to this sink. All the sinks then exchange their cluster size information. Finally, some measures such as standard deviation of cluster sizes are calculated by sinks to detect the dispersion of sensors over sinks. A high standard deviation indicates that the sensors are not uniformly distributed over the sinks in WSN. P-NLB defines two different routing modes which make it an efficient routing in both uniform and non-uniform networks.

First mode is *Smart Shortest Path Mode* (S-SPM) which only uses local information for data dissemination. *Smart* here only means that if a sink has more than one neighboring sensors as candidates for its parent, S-SPM makes the parent selection decision based on some application-specific routing metrics such as energy level, buffer capacity, congestion avoidance, etc. On the other hand, in SPR, parent selection is done randomly form the set of neighbors.

The second mode is not only uses local level information, but also combines it with the clustering information of global level. It has the goal of balancing the network load over all sinks in the networks while also routing data cleverly inside each partition; therefore, it is called *load balancing* mode (LBM). Fig. 5 gives an overview of P-NLB. It shows two routing modes with global and local levels.

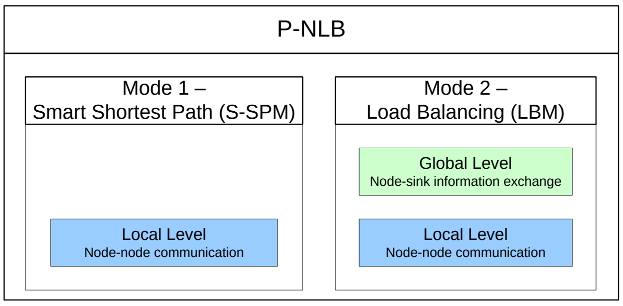

P-NLB
Mode 1 – Smart Shortest Path (S-SPM)
Local Level
Node-node communication
Mode 2 – Load Balancing (LBM)
Global Level
Node-sink information exchange
Local Level
Node-node communication

Fig. 5. Partition-based Network Load Balancing Protocol

4.2 Protocol Structure

In Fig. 4, the setup phase enters the operational phase in ***S-SPM*** or ***LBM***. In the operational phase, sensors must establish dynamic spanning trees rooted at the sinks, which is then used for routing data to the sinks. In P-NLB, the spanning trees are constantly maintained and adjusted to the most efficient routing paths in the network. The state diagram of the sensor and sink nodes in the operational phase is shown in Fig. 6. The steps belonging to the global level have striped boxes, while the steps belonging to the local level have dotted boxes.

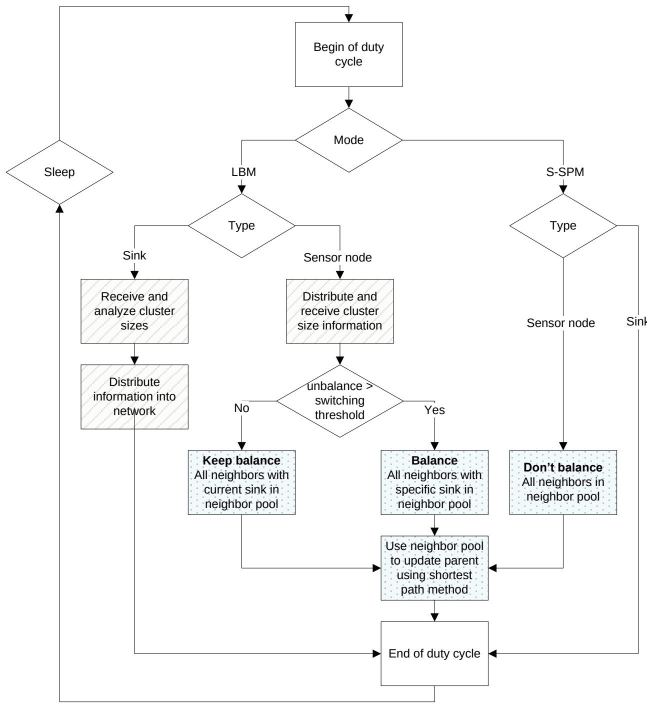
4.3 Global Level – Cluster Information Gathering and Distribution

The goal of the global level is obtaining information about the clusters in the network and providing the sensors with this information. The sinks determine if the clusters in the network are balanced and if that is not the case, they determine which cluster is the smallest. On a local level, the sensors use the information provided by the sinks in combination with local information to make their routing decisions and adjust the clusters. The goal of this technique is spreading the load in the network over all the sinks. The novel part of this technique is that no explicit clustering phase is used, but *the nodes in the network achieve clustering on a global level, by routing on a local level*.

The mechanism of cluster information gathering and distribution has three steps. The algorithm is given in Algorithm I.

- **Information gathering.** Nodes keep track of the number of child nodes they have and aggregate and propagate this information to the sink at the root of the spanning tree. In this way, each sink knows what the amount of nodes in its spanning tree is and thus knows the cluster size.

- **Analyzing.** Assuming (direct) communication between sinks, each sink has information about all the other cluster sizes in the network and consequently the load in the network
- **Distribution.** Sinks distribute this information back into the network, using cross-layer communication. It is a continuous process of gathering, analyzing and distributing the information, there are no specific phases.



Begin of duty cycle
Mode
LBM
S-SPM
Sleep
Type
Type
Sink
Sensor node
Receive and analyze cluster sizes
Distribute and receive cluster size information
Sensor node
Sink
Distribute information into network
unbalance > switching threshold
No
Yes
Keep balance All neighbors with current sink in neighbor pool
Balance All neighbors with specific sink in neighbor pool
Don't balance All neighbors in neighbor pool
Use neighbor pool to update parent using shortest path method
End of duty cycle

Fig. 6. State diagram of operational phase

Algorithm I: Cluster information gathering and distribution

```
N ←set of all nodes;
Desc_i ← set of descendant nodes of node i
C_i ← set of child nodes of node i
P_i ← parent node of node i
S ← set of sinks in network
For each node i ⊆ N do
  //step 1: information gathering
  /* received descendant information of all child nodes, assuming node has any child nodes */
  RXDescChilds(C_i)
  Desc_i = 0
  for each c ⊆ C_i do
       Desc_i = Desc_i + Desc_c
     if i ≠ sink
       /* transmit updated descendant information to parent node, sinks have no parent nodes */
       TXDescParent(P_i, Desc_i)
  else
       //step 2: Analyzing
       /* send own and receive descendant information to / from other sinks
       for each s ⊆ S do
            TXDescSinks(Desc_i, s)
            RXDescSinks(Desc_s, s)
       /* calculate with information of all sinks which cluster is the smallest */
       SC ← CalcSmallestCluster(Desc_s, Desc_i)
  //step 3: Distribution
  for each c ⊆C_i do
       /* send SC to child node, assuming node has any child nodes */
       TXSCChild(c, SC)
End
```

An illustrative example of this two-level routing approach is given in Fig. 7. The first step shows an unbalanced network, with two clusters *A* and *B* of 8 and 21 nodes in each cluster. The arrows show the nodes sending information about the upstream links to the sinks. With this information, a sink determines the amount of nodes in its cluster – the *cluster size*. In step 2, sinks exchange information about each others' cluster sizes and conclude *Cluster A* is the smallest cluster. This information is distributed into the network in step 3. These three steps on the global level are continuously repeated. Step 4 shows the action of the sensors in the network, which happens on the local level. The sensors switch parents, resulting in different sizes of *Cluster A* and *Cluster B* which are now balanced. The balanced network is shown in step 5, with network clusters of 15 and 14 nodes in each cluster. The mechanism in which sensors switch parents on the local level is explained in detail in the next section.

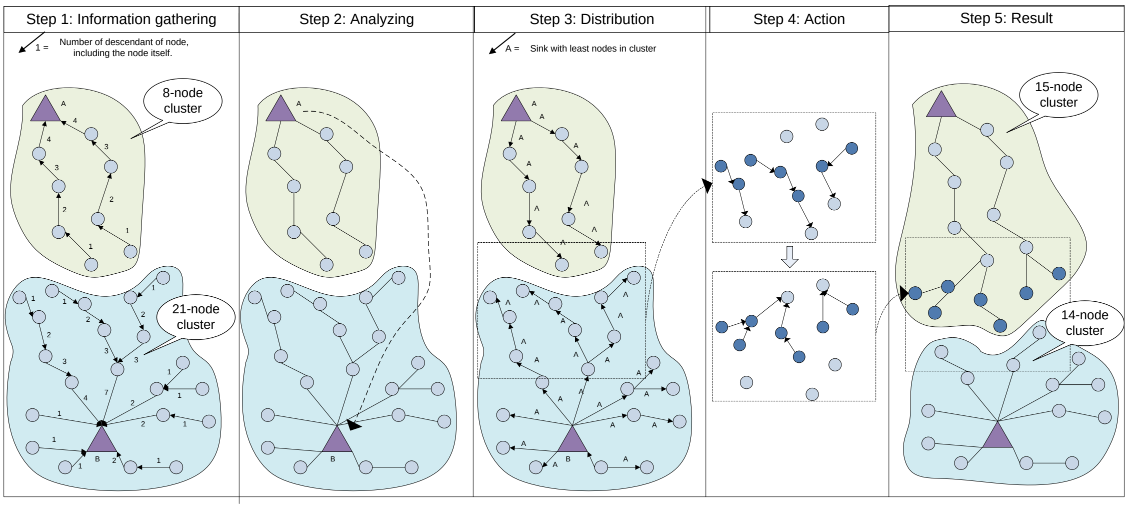


Fig. 7. Overview of global (steps 1-2-3) and local level (steps 4-5) of P-NLB in five steps

4.3 Local level – Optimized Routing Tree Building using S-SPM

On a local level, we use a *metric-based routing* mechanism where every sensor decides for itself what the next step should be to create an efficient routing path and forward data over it. The selected next-hop neighbor is called the *parent node* and the forwarding node itself is called the *child node*. Instead of the SPR's *randomized parent selection* of any neighbor closest towards a sink, P-NLB uses well defined metrics to increase the efficiency of this method.

The parent selection depends on the clustering information provided by the sinks on the global level – in case of the balancing mode – and local information and the *routing metric* of the node. The routing metric of a sensor depends on the demands of the application running on the WSN. The routing metrics are:

- ***Child nodes.*** Select neighbor with the smallest number of child nodes.

- ***Descendants.*** Select neighbor with the smallest number of descendants – upstream nodes.
- ***Remaining Energy level.*** Select neighbor with highest remaining energy level; thus, nearly depleted nodes are avoided in the routing path.
- ***Buffer.*** Select neighbor with the least amount of nodes in the packet queue; therefore, congestion and latency can be reduced.

We will now take a closer look at how the adjustments on the spanning trees in step 4 of Fig. 7 are performed. This process consists of four steps, one step for defining the *neighbor pool* and three steps for selecting a parent from that neighbor pool. The algorithm is given in Algorithm II.

Using global and local information to define neighbor pool: In large scale dense sensor networks, every node has several neighbors, which can all be selected as the parent node for forwarding the data. In P-NLB, each sensor node creates a *neighbor pool* which contains only the neighbors meeting the terms of the routing mode the node is in. In *S-SPM*, all neighbors are part of the neighbor pool. In *LBM,* information about cluster sizes is provided by the global level and this separates the neighbors into two categories:

- Neighbors that are in the same cluster as the node
- Neighbors that are in the cluster which has the smallest cluster size

In order to prevent constant switching between clusters of nodes that are right between two sinks – network *oscillation* – nodes have a threshold which helps them decide to switch to another cluster or stay in its current cluster. In small networks this oscillation doesn’t have much effect, but in larger networks with more nodes between two clusters preliminary simulation has shown it causes highly instable routing paths and results in decreased routing performance. In order to counter this oscillation, the parameter *switching threshold* – defined as a certain amount of nodes – is introduced, which stops nodes from attempting to balance slightly unbalanced networks. A node will not attempt to balance a cluster if the unbalance is smaller than the *switching threshold*. In *LBM*, this parameter is used for determining the correct neighbor pool.

Defining the neighbor pool is done in one step in *S-SPM* and two steps in *LBM*:

- **Step 1a.** Both modes: get all one-hop neighbors.
- **Step 1b.** In LBM only: remove neighbors from certain cluster depending on the *switching threshold*.

Using local information and routing metric to select parent from neighbor pool: Now nodes have composed the correct neighbor pool, and it is time for them to actually select a parent from neighbor pool and build the spanning tree in the network step by step as explained as follows:

- **Step 2.** Check hop count of neighbors; only consider neighbors with the lowest hop count in next steps.
- **Step 3.** Apply routing metric on the remaining neighbors. If routing metric is *Child Nodes*, it only keeps the neighbors with the smallest amount of child nodes. If routing metric is *Buffer*, it keeps only the neighbors with the least amount of packets in their buffers, etc.
- **Step 4.** All neighbors left have the same properties and one random neighbor is selected as the parent.

Algorithm II: Neighbor pool construction and parent selection

```
N ←set of all nodes;
$P_i$ ← parent node of node i
$NB_i$ ← set of neighbors of node i
$NBP_i$ ← neighbor pool of node i
node i ⊆N
U ← Cluster size unbalance
ST ← switching threshold
Cli ← cluster node $_i$ belongs to
CS ← smallest cluster
//step 1: neighbor pool construction
if routing mode == shortest path
   /* all neighbors are in neighbor pool
   $NBP_i$ = $NB_i$
else
     /*routing mode is balancing, check switching threshold*/
     for each nb ⊆$NB_i$ do
          if ST > U
               /* balance clusters*/
               if $CL_i$ == CS
                 AddNbrToNBP($NBP_i$, nb)
          else
            /*stay in same cluster, don’t change clusters */
            if $CL_i$ == $CL_{nb}$
               AddNbrToNBP($NBP_i$, nb)
End else
//step 2: Check hop count, discard neighbors which have not the lowest hop count */
$NBP_i$  ← CheckHC($NBP_i$)
//Step 3: Apply metric on neighbor pool
$NBP_i$  ← ApplyMetric($NBP_i$)
//Step 4: Parent selection
$P_i$ ← SelectParent($NBP_i$)
End
```

The example in Fig. 7 is used to show both neighbor pool construction and parent selection. The partial network is shown in Fig. 8. In this example, none of the nodes have updated their parents yet, so the network is still unbalanced. *LBM* is used in combination with routing metric *Buffer*. In Step 1a *Node 1* finds out it has six neighbors. In Step 1b it removes the four neighbors from the neighbor pool which are not in the smallest cluster and is left with a neighbor pool of two nodes. It has now constructed its neighbor pool and will continue with selecting the parent. In Step 2 both *Node 2* and *Node 3* have the same hop count, but *Node 3* has a smaller packet queue size and therefore *Node 3* is selected as parent in Steps 3 and 4. The other nodes do the same and the result (in the last drawing of Fig. 7) is that by routing on local level the clusters at a global level have changed. Table 2 contains the local information *Node 1* has about its neighbors; the global information is that *Cluster A* is the smallest cluster in the network.

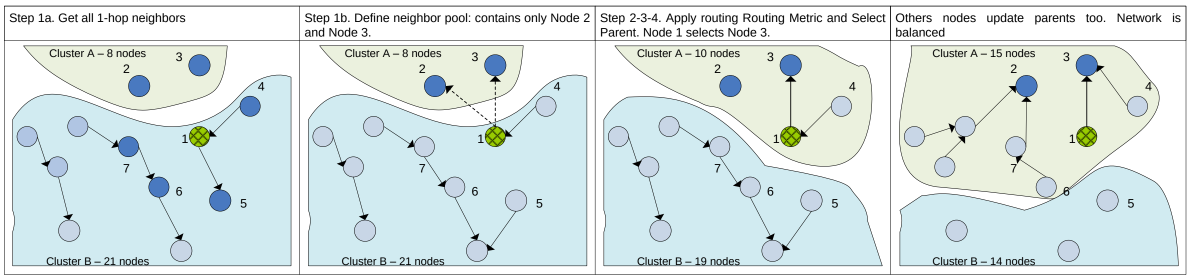

Step 1a. Get all 1-hop neighbors
Step 1b. Define neighbor pool: contains only Node 2 and Node 3.
Step 2-3-4. Apply routing Routing Metric and Select Parent. Node 1 selects Node 3.
Others nodes update parents too. Network is balanced
Cluster A – 8 nodes
Cluster B – 21 nodes
Cluster A – 8 nodes
Cluster B – 21 nodes
Cluster A – 10 nodes
Cluster B – 19 nodes
Cluster A – 15 nodes
Cluster B – 14 nodes

Fig. 8. Steps of local level of P-NLB

Table 2. Balancing parent select example: local neighbor information

Neighbor	Cluster	Hop Count	Routing Metrics			
			Child Nodes	Buffer	Energy Level (%)	Descendants
2	A	4	0	3	80	0
3	A	4	0	1	71	0
4	B	4	0	8	56	0
5	B	2	0	5	88	2
6	B	2	1	3	92	2
7	B	3	1	1	57	1

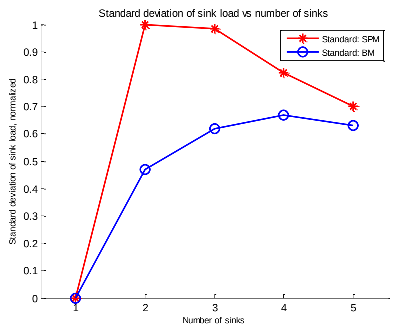
Loop detection and avoidance: In *S-SPM*, loops are not an issue, because due to the shortest path paradigm loops cannot be created in the network. On the other hand, in *LBM*, precautions are needed to detect and avoid loops in the network. In P-NLB, loops are caused due to outdated local information about neighbors. Nodes change their parent constantly; therefore, the routing paths in the network change. However, it takes some time for this information to reach all the nodes on the routing path and the neighbors of these nodes. We use the technique of tracking *routing hop counts* to detect loops in the network. If a nodes detects a loop in the routing path, the path is broken and a new (loop free) path is established. In that case it will set a back-off timer and while this timer counts down to zero, the node is able to receive updated information about its local neighborhood. When the back-off timer reaches zero, it will again select a new parent.

5 Simulation Results

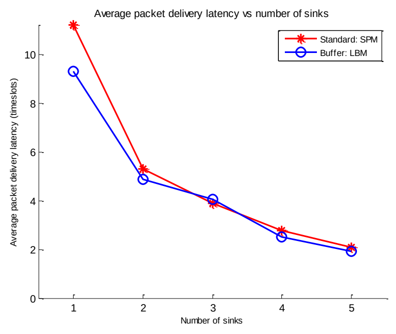
We established 200 random connected networks consisting of 64 nodes and 2 sinks in the 'routing metrics performance simulations' and 1 to 5 sinks in the 'multi-sink performance simulations' in Matlab. These simulations show the performance of load balancing and routing algorithms in such random deployment situations. The simulations is run for 5.000 MAC frames, so each sensor has 5000 opportunities of performing some action i.e. generating and sending data. Every sensor generates 1 packet every 6 frames. The performance of the network is measured using the following performance metrics: *(i) Average packet delivery latency:* The end-to-end delay from the time the packet is generated at the sensor node until it arrives at a sink, *(ii) Network* *lifetime:* Time from initialization till the first network partition (due to energy depletion) occurs, *(iii) Packet Delivery Ratio:* Packets delivered at the sinks as percentage of the total packets generated by the sensors. Best effort routing is used which means no resending of lost packets, and *(iv) Load balance:* Standard deviation of load on sinks in the network.

5.1 Multi-sink performance

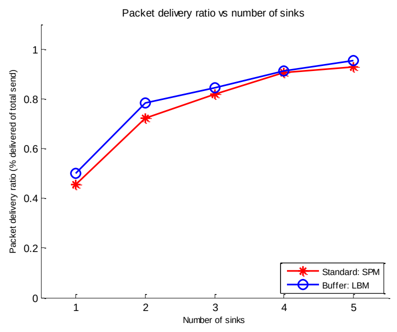
When looking at the influence of the number of data sinks, Fig. 9 shows both routing modes benefit on all areas from an increasing amount of sinks in the network. *Balancing mode* in combination with the *Buffer* routing metric is much better able to uniformly distribute the load over all sinks than *shortest path mode* as shown in Fig. 9(a). The advantage is the largest with two sinks in the network, but decreases when more sinks are added to the network, because both cluster sizes and average path lengths between sinks and nodes decreases, when there are more sinks in the network. Although LBM leads to longer routing paths, the reduced congestion compensates this, resulting in a slightly lower average latency in Fig. 9(b). The reduced congestion leads also to less packet drops in congestion nodes, and consequently in Fig. 9(c) the packet delivery ratio is higher, although not completely in accordance with the better load balancing of BLM. An explanation for this behavior is given in the next section. LBM in combination with the *Energy level* routing metric prevents overburdening nodes close to the sinks with data, resulting in a higher network lifetime in Fig. 9(d).



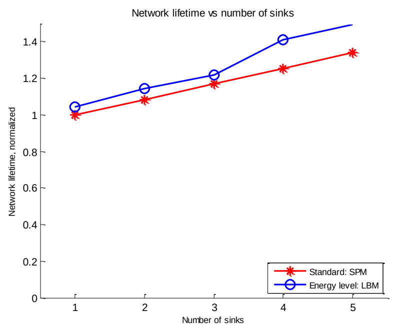
(a) Load balancing



(b) Average packet delivery latency



(c) Packet delivery ratio

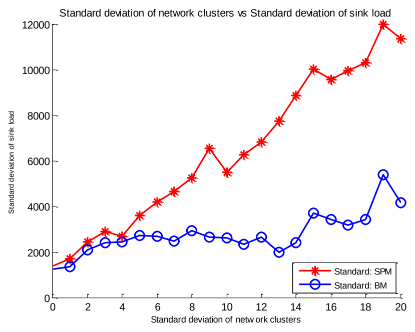


(d) Network lifetime

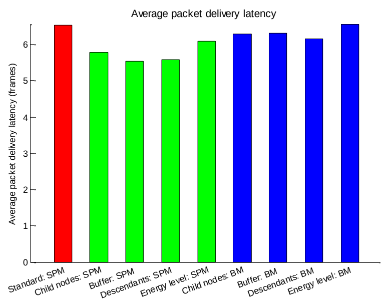
Fig. 9. Multi-sink performance simulation results

5.2 Routing metric performance

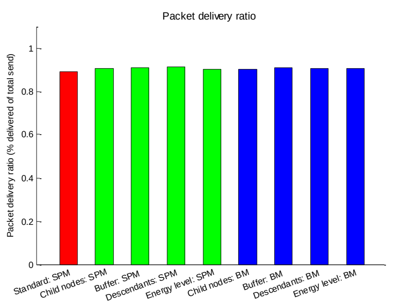
In these simulations, all routing metrics are used on each of the 200 random connected networks. As shown in Fig. 10 (a) an increasing difference between initial cluster sizes in the network also leads to an increase in a difference of the load on the sinks when using shortest path routing. In *LBM* this increase is very limited compared with *S-SPM*, only with a very high standard deviation the sink load deviation starts to increase. The routing metrics performances show that in general latency (Fig. 10 (b)) is higher in balancing mode, compared with *S-SPM*. The source of this extra latency is the increased routing paths lengths of the *LBM*, which is not compensated enough by the reduced congestion. Packet delivery ratio (Fig. 10 (c)) is equal in both routing modes, which is not as expected considering the fact that the difference between the loads on both sinks is much higher. One reason for his result is that congestion in the network is not only situated near and in the sinks, but also locally on other places in the network. Local bottlenecks are found in the network where a few number of nodes has to process data of a large uplink cluster consisting of many descendant nodes. By using the LMAC as underlying MAC protocol bandwidth is divided equally over all nodes, therefore those congested nodes has as much bandwidth as uncongested nodes, while those congested nodes need more bandwidth. So load balancing does divide the absolute load better over the sinks in the network, but congestion also occurs locally within the network and balancing the load over the sinks does not solve that problem. When we look at the performance of the routing metrics, we see that routing metric *Buffer* performs best in both latency and packet delivery ratio. This routing metric is most congestion aware and is able to minimize latency. As expected, the network lifetime performance metric (Fig. 10 (d)) is highest when using routing metric *Energy level*, especially in *S-SPM*.



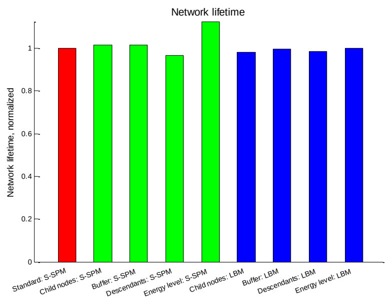
(a) Load balancing



(b) Average packet delivery latency



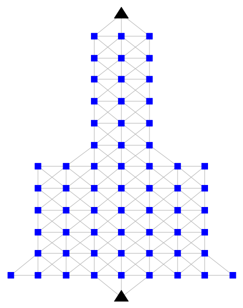
(c) Packet delivery ratio



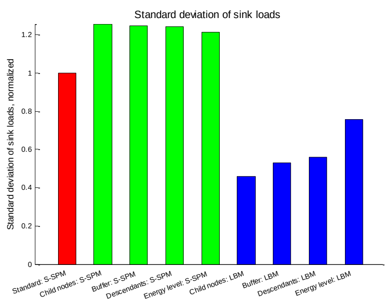
(d) Network lifetime

Fig. 10. Random topology simulation results

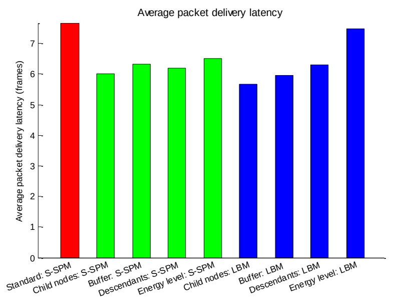
In order to further investigate the packet delivery ratio performance issue, we also used the same routing metrics for simulations on a network with a custom topology, which features a small and a large cluster. This custom topology is much more regular and therefore does not contain the local bottlenecks as mentioned before. Simulation results in Fig. 11 show that in such regular topology load balancing in BLM is much more effective; both latency and packet delivery benefit from the uniformly distributed load.



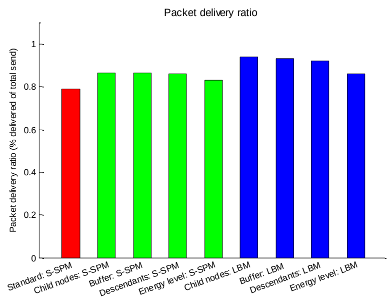
(a) Custom topology



(b) Load balancing



(c) Average packet delivery latency



(d) Packet delivery ratio

Fig. 11. Custom topology simulation results

6 Conclusion and Future Work

In this paper we presented P-NLB, a routing protocol for large-scale multi-sink WSN, with uses a global load balancing and clustering technique with local metric-based routing for optimized routing tree building. On the global level information about cluster sizes in the network is gathered by sinks and distributed to the sensors. Its distributed approach results in very low communication overhead due to the use of cross-layer information of the MAC layer. Except for a one-time broadcasting for detecting initial cluster sizes, as part of the LMAC setup phase, only local information exchange is used, making it very scalable to large sensor networks.

Simulations show that the P-NLB *balancing mode* uniformly distributes the load efficiently over the sinks in the network. In random network topologies this results in a higher latency, caused by longer routing paths. Packet delivery ratio does not always benefit from balancing the load; *balancing mode* gains most advantage compared with shortest path mode when the initial difference between clusters' sizes increases. Routing metric of *Buffer* results in the lowest latency and highest packet delivery ratio of all routing metrics. Routing metric of *Energy level* results in the highest network lifetime. Although the load in the network is more balanced using *balancing mode*, latency and packet delivery ratio does not reflect this. A source for that is that congestion occurs sooner in nodes *around* the sinks than the sinks themselves.

As a future work, we will focus on the impact of mobility of both sensors and sinks on the performance of P-NLB. We expect that the flexibility of P-NLB in adjusting the routing trees in the network makes it well suited for mobile sensor networks.

References

1. ICT-FP7 SENSEI project, "Integrating the Physical with the Digital World of the Future", project website: http://www.ict-sensei.org/ (last access 01.12.08)
2. Bejerano, Y., Han, S.J., Kumar, A.: Efficient Load-Balancing Routing for Wireless Mesh Networks. In: Computer Networks, Vol. 51, No. 10, pp. 2450--2466 (2007)
3. Gao, J., Zhang, L.: Load Balanced Short Path Routing in Wireless Networks. In: IEEE INFOCOM, pp. 1099--1108. (2004)
4. Zhang, Y., Huang, Q.: A Learning-based Adaptive Routing Tree for Wireless Sensor Networks. In: Journal of Communications, Vol. 1, No. 2, pp. 12--21. Academy Publisher (2006)
5. Gupta, G., Younis, M.: Load-balanced clustering of wireless sensor networks. In: Proceedings of the IEEE International Conference on Communication, Vol. 3, pp. 1848--1852. May (2003)
6. Chatterjee, P., Das, N.: A Distributed Algorithm for Load-Balanced Routing in multihop wireless sensor networks. In: 9th ICDCN, LNCS 4904, pp. 332—338. (2008)
7. Dai, H., Han, R.: A Node-Centric Load Balancing Algorithm for wireless sensor networks. In: IEEE GLOBECOM, pp. 548—552. (2003)
8. Raicu, I., Schwiebert, L., Fowler, S., Gupta, S. K.S.: Local Load Balancing for Globally Efficient routing in wireless sensor networks. In: International Journal of Distributed Sensor Networks, Vol. 1, pp. 163–185. Taylor & Francis Inc. (2005)
9. Low, C.P., Fang, C., Ng, J.M., Ang, Y.H.: Efficient Load-Balanced Clustering Algorithms for wireless sensor networks. In IEEE ICC, pp. 3485—3490 (2007)
10. Zhou, C., Krishnamachari, B.: Localized Topology Generation Mechanisms for Wireless Sensor Networks. In: IEEE GLOBECOM, pp. 1269—1273. (2003)
11. Hoesel, L.F.W., Havinga, P.J.M.: Design Aspects of An Energy-Efficient, Lightweight Medium Access Control Protocol for Wireless Sensor Networks. In Int. Journal of Communication Systems. 2006, pp. 1--21