Resource Management Tools

Anh Nguyen
Supervisor: Corinna Schmitt
Seminar: Innovative Internet Technologies and Mobile Communication WS 12/13
Chair for Computer Network Architectures and Services
Department for Computer Science, Technische Universität München
Email: anh.nguyen@tum.de

ABSTRACT
Sensor networks perform and are used in many different kinds of fields, but are especially given tasks, which are difficult and almost impossible for a human to do. Besides monitoring the environment and collecting necessary information, sensor networks have to manage and supply themselves with essential resources to perform the tasks that were allocated to the whole network.
This paper focuses on resource management and the development of effective resource management tools, which has to face many challenges due to resource limitations, such as energy consumption and scalability. Peloton and Tiny Network Manager cover up most of these challenges with different approaches. While Peloton uses a ticket abstraction to distribute allocations and perform optimization on the whole sensor network process, the Tiny Network Manager has an implemented "request-reply-mechanism" to manage its assignments and resources.

Keywords
Wireless sensor network, resource constraints, resource management tools, Peloton, Tiny Network Manager

1. INTRODUCTION
Sensor networks have been developed from innovations in wireless communication within the past few years and are one of the most important technologies in the 21st century. Nowadays sensor networks are used in military for communication and targeting, in health systems for monitoring and assisting (disabled) patients, and other application fields, such as managing inventory in business, and monitoring inaccessable and disaster areas.[1]

Figure 1 shows an example of a sensor node structure with its units and their connection to each other: The basis of the sensor node is its power unit, which keeps the sensor node and its units alive. The overall task of a sensor network is to monitor physical or environmental conditions through the sensing unit, which monitors the environment and converts the analog signals to digital ones, and finally passes it on to the processing unit. After performing quick local data processing, all the collected data will be passed through the network and routed back to the sink with a multi-hop architecture.[1]

Since sensor nodes are often deployed in inaccessable or disaster areas, it is common that they have to move their position from time to time due to unconventional events and

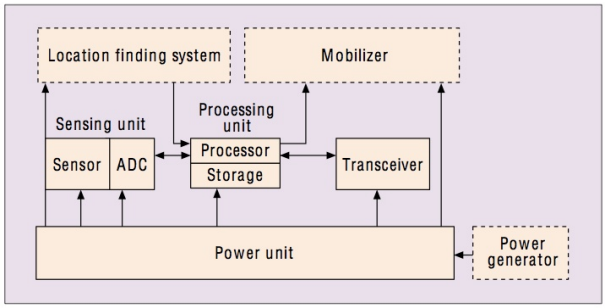

Figure 1: Sensor node structure [1]

situations. Therefore a location finding system and a mobilizer are often implemented as additional units.[1][2]

Wireless sensor networks are intended to perform for "long periods of time, over relatively large physical spaces, and in places that are difficult for people to reach."[2] Since sensor nodes in wireless sensor networks lack in terms of e.g. energy, operating systems and tools were developed to manage exhausting resources and constraints of the sensor nodes. Those tools are called resource management tool and help managing and distributing the resources to all the sensor nodes in a network.[1][3]

In the following section definitions are given on resource management and resource management tools and the challenges during the development of resource management tools in general, as well as the chosen resource management tools Peloton and Tiny Network Manager are described afterwards. In comparison of those resource management tools in section 3, their different and common features are characterized. Lastly a conclusion is given and a future outlook for sensor networks and resource management tools in technology is presented.

2. CHALLENGES DURING THE DEVELOPMENT OF RESOURCE MANAGEMENT TOOLS

2.1 Resource management functions
According to Tenrox, resource management is defined as an "efficient and effective deployment and allocation of an organization's resources when and where they are needed."[11] In terms of sensor networks it is about how to save the necessary resources and optimize their consumption for longer

doi: 10.2312/NET-2013-02-1_10

availability. Moreover resource management is not only about lowering the resource consumption, but also to optimize the communication between each node in the sensor node to distribute tasks and thereby the resource allocations effectively. The result of an efficient management process would be optimized and distributed allocation of resources, so that each single node is not overloaded with assignments, but is still kept busy. It is an efficient employment of sensor nodes to effectively perform and succeed their assigned tasks.[11]

2.2 Challenges and difficulties

Sensor networks, on the one hand, usually have stable sensor nodes that can withstand the harsh and uncertain environment and avoid physical damage, but lack in essential resources on the other hand. Moreover their disadvantage is that each sensor node can only collect and process data, but not communicate and share the collected information and also information on their node state with other nodes. Therefore resource management tools are used to compensate this and optimize all processes in the whole sensor network.[3]

The challenges in developing a resource management tool therefore is to implement efficient processes and algorithms to manage scarce resources of the sensor nodes, such as the minimal life-span, limited memory and stack space and limited radio bandwidth[1][2][5]:

- Minimal life-span
  Power consumptions of each sensing node have to be kept low, since energy is the scarcest resource of a sensor node. The resource management tools therefore aim for the optimization of power usage and with that also the maximization of a node's lifetime, so that sensor nodes are able to work for a long time and are prevented from a failure or breakdown in an early state.[2][9]

- Limited memory and stack space
  The memory and stack space of a sensor is limited and as a consequence the amount of data collection and processing as well as the amount of data transfers are also limited. If too much data is processed and transferred it will result in a stack overflow, causing the process or the node to crash at the worst. Therefore the management tool has to define and rank the data in terms of importance and consider which data to process first or which data to drop.[3]

- Limited radio bandwidth
  Besides having limited stack space, the bandwidth of wireless sensor networks is also limited and much lower than that of a wired sensor network. In this case the resource management tool has also to decide which data to transfer or which data to keep in the queue.[2]

Besides managing sensor node resources, the implemented processes have also to meet the quality standards through ensuring fault tolerance, the scalability of the whole system as well as real-time performances and the reliability of the collected and processed data, which also have a strong impact on the whole sensor network process[1][2][5]:

- Fault tolerance
  The term fault tolerance refers to the reliability of sensor nodes. In case of failure or a breakdown of single nodes due to lack of power, interruption or damage, the resource management tool has to ensure that the task of the broken sensor node and the overall task and functionality of the whole network are not be affected.[1]

- Scalability
  Sensor networks are composed of a huge amount of sensor nodes. Depending on the usage its scale can reach up to thousands or millions of sensor nodes. Since sensor nodes are not able to think and work autonomously, the resource management tool has to coordinate the communication and interaction of sensor nodes and other hardware components to perform tasks faster and more effective, and maintain a manageable structure of the sensor node.[1]

- Real-time performance
  Many sensor network applications have to stick to a certain time schedule to interact with other sensor nodes and hardware components. Moreover the assigned tasks have to be completed until a certain deadline. The resource management tool has to ensure real-time behavior through efficient management of the tasks and sensor nodes also in regard of the available resources of each sensor node. Constructing and implementing an algorithm to ensure real-time performance would be quite a challenge because of the dynamic environment.[3]

- Reliability
  The term reliability in this case refers to the collected information. Since the sensor networks are used to collect important data and information, the resource management tool has to make sure that the processing of collecting necessary information is always reliable and transferred to the correct places. Losing important information would lengthen the whole sensor network process duration or in the worst falsify the statistics like incorrect information.[1]

If we want to increase the functional performance of a wireless sensor network, we have to intelligently manage not only the limited resources, but also the interaction between each sensor node in the system.[5]

Until now a couple of resource management tools, such as TinyOS, Pixie, SOS and Eon, have been developed: They only focus on managing resources for individual nodes, although the coordination of resource management decisions across the whole network would be more efficient and accurate.[4]
This approach is not advisable, because the nodes in a network always have to interact with other nodes and hardware components. The sole management of a sensor network as a collection of independent sensor nodes would not solve the problems of limited resources and network constraints. On the contrary: It is necessary that nodes share information on their local state and communicate and collaborate with other nodes to assign tasks to achieve the most efficient use of resources and task processing.[2][5]

doi: 10.2312/NET-2013-02-1_10

As a result, the final tool should be able to support and enable "low latency, energy-efficient operation, built-in autonomy and survivability, and low probability of detection of operation"[3].

3. RESOURCE MANAGEMENT TOOLS

3.1 Peloton

One of those resource management tools is called Peloton and was developed in 2009 as a "new distributed sensor OS".[4] Peloton builds up on Pixie, a node-level operating system for sensor nodes to manage and control over resource availability: Pixie works with resource tickets, which are sent by and to the operating system. A ticket represents a time-bounded right to consume a certain amount of a resource and is a "flexible currency for resource management within the node".[4] In this way, Pixie can respond to the resource ticket allocations and distribute available resources.

Peloton has borrowed the Pixie's idea of using resource tickets and extended it to a new ticket abstraction, which has implemented resource management mechanism called vector tickets, distributed ticket agents and state sharing. But unlike Pixie and also other operating systems, such as TinyOS and EON, which manage the resources on individual nodes only, Peloton is not focusing on individual nodes but the cooperation of all sensor nodes for resource management.[4]

3.1.1 Ticket Abstraction

It has been already mentioned that Peloton has implemented its own ticket abstraction through extending Pixie's resource ticket mechanism. It consists of the vector tickets, distributed ticket agents and a state sharing system:

Vector Ticket is a programming abstraction, which represents the right of a sensor node to consume resources for performing operation. A vector ticket V consists of resource tickets T_i. Each vector ticket is displayed as a tuple (n, R, c, t_e) that represents the "time-bounded right to consume up to c units of resource R until expiry time t_e at node n"[4]. Moreover it can capture the complete resource allocation of an operation of several nodes and is also used for tracking and controlling this allocation in the network. Therefore vector tickets record the consumed resources of the sensor nodes and provide feedback to the application in terms of resource availability and usage. One strength of the vector tickets it that they can be decoupled if the allocation is not necessary anymore and give resources free for other needy nodes.[4]

The **distributed ticket agent** mechanism permits resource management decisions all across nodes, clusters and the network as a whole. They manage vector tickets, track resource availability of single nodes as well as of a set of nodes and distribute resource allocations and also perform resource management policies across all the nodes. Because of distributed ticket agents and their functions, resource allocations can be performed individually by nodes, collectively by a group of nodes, or globally by a base station. In the resource allocation process a node must either acquire a vector ticket from the node's local ticket agent or from a third-party agent, such as the base station or another node in the network, to get a resource allocation. But even if a ticket is acquired, it is possible that it may be revoked before its expiry time, because of changing conditions.[4]

The **state sharing** mechanism is in charge of sharing node states on local resource availability so that efficient coordination among all the sensor nodes is possible. For this efficient coordination among all the sensor nodes in the network, Peloton has build-in mechanisms for node sharing within the neighborhoods, clusters and across the whole network. Peloton's API makes it possible for nodes to share information on local resource availability into a shared tuple state. This information on the other hand can be called up from the shared tuple space to make it readable for each other node. Those data are refreshed frequently, but only between nodes, that are direct neighbors. Information from distance nodes is refreshed less often, but it is always possible to request a direct update from another node to obtain its latest state. Those state sharing operations consume energy and bandwidth and thus, also requite resource tickets.[4]

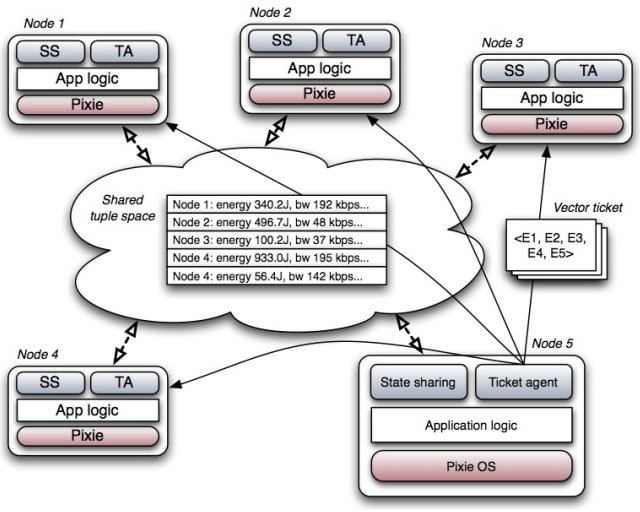

Figure 2: Peloton network with five nodes[4]

Figure 2 shows a Peloton network with five sensor nodes. Each node has the same structure: They are build up on the Pixie operation system and extended with an application logic and the ticket agent and state sharing mechanism, that enable the nodes to make resource requests and send all their vector tickets to other sensor nodes. Moreover all of the nodes are linked to the shared tuple space through state sharing and can provide and access information on the local node state of each node in the network.[4]

3.1.2 Resource Management

Besides the ticket abstraction, which supports the distribution of resource allocations in the sensor network and optimizes the overall sensor network process, Peloton has implemented adaptive cluster-based routing, duty cycling and reliable data collection as further resource management mechanism[4]:

Adaptive cluster-based routing is a frequently used method for energy-efficient routing in sensor networks: One routing and communication protocol for wireless sensor networks is

doi: 10.2312/NET-2013-02-1_10

called “LEACH” and stands for “Low Energy Adaptive Clustering Hierarchy”.[6]
LEACH works with cluster architecture and its main idea is to collect data from distributed sensors and transmit it to a base station. In LEACH some nodes can elect themselves as clusterheads. As clusterheads they are responsible for receiving data and data packets from other clusterheads and forwarding it to members of its cluster or to the base station. Since the data transmission to the base station requires and consumes too much energy the clusterheads “rotate”, which means that the clusterheads take turns in being the clusterhead. Through this rotation huge energy consumption can be avoided and a longer lifetime of the sensor node can be ensured. Moreover the clusterhead selection can be done on local information, which does not need a communication with the base station and other entities outside the network and therefore reduces the energy consumption as well.[6]
Peloton has an implemented variant of LEACH, in which a sensor node can elect itself as a clusterhead based on the energy consumption profile of neighboring nodes. Each clusterhead temporarily becomes the ticket agent for the cluster members. In this position it can assign vector tickets to manage the overall communication and resource consumption of the cluster members.[4]

Duty cycling the one of the most common forms of resource management.[4] The idea of duty cycling is based on the assumption that nodes are not used for communication and transmission issues all of the time. With this assumption, Peloton has build in mechanism that allows sensor nodes to change their active state into an idle one and vice versa to save up energy consumption. Therefore the sensor nodes are only used when it is necessary. It is often difficult to determine an appropriate duty-cycling schedule statically as the implementation and usage of duty cycling can affect data fidelity, network connectivity and also sensor coverage. But with a precise knowledge about the network topology and transmission structure, it is possible to coordinate and determine the sensor node schedules over the whole network.[4][7]

Another resource management function that is enabled in Peloton is the **reliable data collection** of high data-rate signals.[4]
Reliable transfer requires substantial bandwidth and sufficient energy resources. Sensor networks lack in both bandwidth and power in the attempt to acquire high data-rate and high-fidelity signals throughout the whole network. As a result only the most “interesting” or most “prominent” signals will be acquired, while the rather “uninteresting” signals are left out. Therefore the optimization of high data-rate signal collections would benefit the resource management in sensor networks.[4][8]
One approach to do so is to enable clusterheads in the network in a similar way as in LEACH: Clusterheads can manage the stored data of neighboring nodes and perform local optimization to rank the importance of the signals. The signals with the highest ranking or priority should be provided with energy and bandwidth resources. With the coordination of the clusterheads, the transfer of the high data-rate signals can be managed and scheduled to the other nodes or to the base station. This approach of reliable data collection therefore allows a reliable transfer of high data-rate signals without consuming much energy and causing a communication overhead. The maximization of reliability of sensors maximizes the reliability of the completion of the assigned tasks, and thus leads to an overall solid performance.[4]

With all those resource management mechanism, Peloton is able to efficiently distribute resources within the network. The combination of Peloton’s reliable data collection with the node energy schedule through duty cycling is not perfect yet, but make a good basis for the development of collaborative applications, which handle changing node states.[4]

3.2 Tiny Network Manager

Another approach for optimizing resource consumption and high performance is called Tiny Network Manager. Tiny Network Manager, short TNM, is a resource management tool that is developed in 2010, based on devisable management, which is a kind of “autonomous management, where different network managers detect network events and do the necessary tasks based on network resources, predefined policies, intuition and intelligence”.[5]

Two different kinds of Tiny Network Manager applications exist:
If Tiny Network Manager is used in large-scale sensor networks, i.e. a sensor network with a huge amount of sensor nodes, the software will reside and control the resources from the clusterhead nodes by collecting and analyzing information from the cluster members, whereas the Tiny Network Managers of the inner cluster members communicate each other to handle uncertain events.
If Tiny Network Manager, on the contrary, is used in a flat wireless sensor network architecture, the software will manage the resources from a base station through the Internet to handle uncertain events[5].

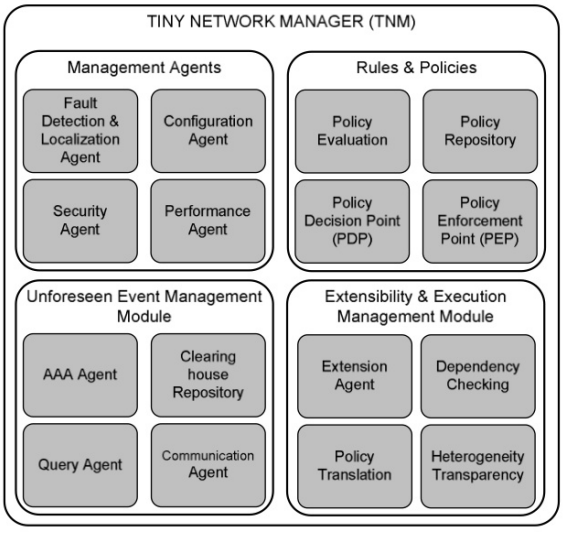

Figure 3: Tiny Network Manager modules[5]

In Figure 3 the Tiny Network Manager structure is displayed, which is composed of the four modules Management Agents, Rules and Policies, Unforeseen Event Management and Extendibility and Implementers, whereas each of them has its own additional components as well:

doi: 10.2312/NET-2013-02-1_10

Management Agents are in charge of detecting changes in the node collection and have specific management tasks such as configuration management, fault management and performance management.

The module **Rules and Policies** and its sub-modules provide policy based management, which makes autonomous management possible.

Unforeseen Event Management modules are trying to look for a solution if specific network state changes come up, which are impossible to handle with the existing rules and policies.

The **Extendibility Module** manages all management policies, agents and functions and can include, remove and modify them if it applies.

3.2.1 Entities

In the resource management process, Tiny Network Manager does not work on its on but with the collaboration of the following entities:

- Network Manager
- Clearing House
- Resource Manager

The **Network Manager**
is an entity that works at the WSN gateway and is composed of the TNM and the Resource Management Module.[5] The TNM performs its usual tasks, which are already described and explained in the previous chapter. The Resource Management Module carries out three main tasks - monitoring, managing and resolving – that are supported by their respective modules and sub-modules, which can be seen in Figure 4:

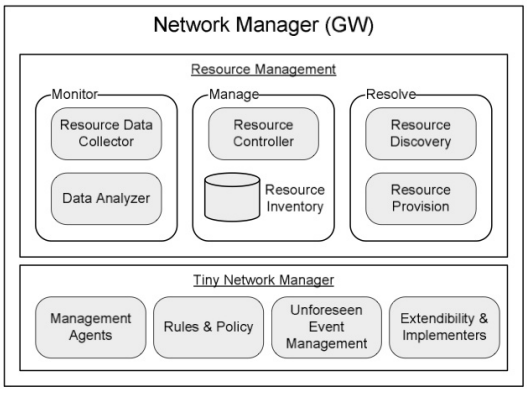

Figure 4: Network Manager[5]

- The monitoring module is composed of the Data Collector module, which collects resource information from the sensor node, and the Data Analyzer, which measures and evaluates the resource information.
- The managing module consists of its main maintenance entity Resource Controller and the database Resource Inventory, which stores all the collected resource information.
- The resolving module is in charge of finding new resources or alternative ones through Resource Discovery and implements it through the Resource Provision sub-module.

With the AAA Agent in the TNM module, the Network Manager can maintain secure sessions to the Clearing House and the Resource Manager.[5]

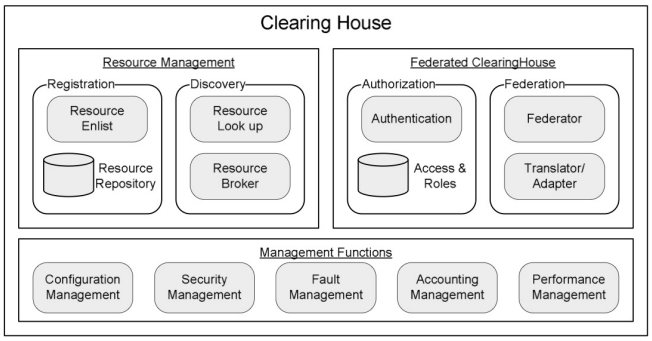

Figure 5: Network Manager[5]

The **Clearing House** (CH) works in the sensor network and consists of a Resource Management Module, the Federated Clearing House and Management Functions. It manages all the information about resources and can provide the location and connection mechanism of registered resources if it is requested.[5] Figure 5 shows the structure of the CH: Same as the Network Manager it has a module to store and maintain all resource information in a database. The Discovery module is the main entity of the CH and handles all the resource requests, whereas the module Management Functions is used for internal management and assistance. The Federated Clearing House is responsible for maintaining all Clearing Houses and providing authorization for communications between CH-CH, CH-RM, and CH-Gateway (Network Manager).[5]

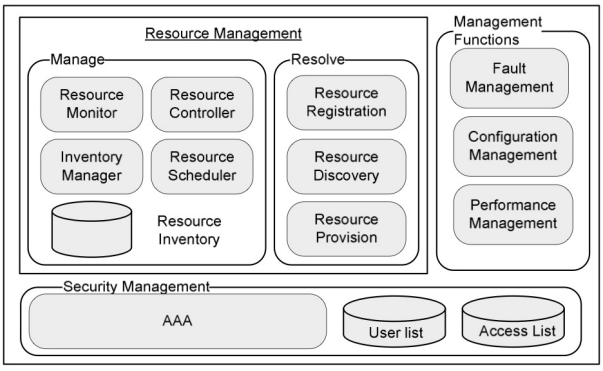

Figure 6: Resource Manager[5]

doi: 10.2312/NET-2013-02-1_10

The **Resource Manager** (Figure 6) is the last entity, which collaborates in the TNM process, and is in charge of resource management in the sensor network.[5]
It has two main modules, Manage and Resolve, and its submodules, which perform the management process that will be described in the further paper. Moreover it provides the same Management Functions module as the CH and a Security Enabler, which enables the interaction between CH-RM, and CH-Gateway.[5]

3.2.2 Resource Management

The Resource Manager performs the resource management process in collaboration with TNM, the Network Manager and the Clearing House: The Resource Management module as well as the Data Collector and the Analyzer monitor the resources. If a resource's performance is decreasing, the Resource Controller has to look for an alternative resource. If no solution can be found, then the Request-Reply-Mechanism will be initiated, which works in the following way[5]:

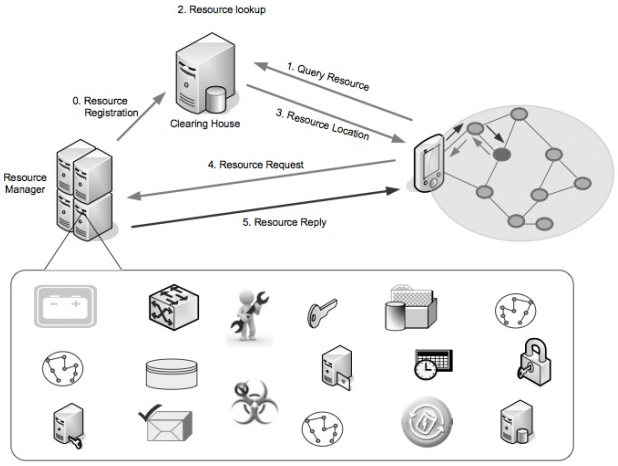

Figure 7: Request-Reply-Mechanism[5]

1. The Resource Discovery module establishes a connection to the Clearing House and sends a resource request

2. At the Clearing House, the Authorization module hosts this request establishes a new session

3. After the session is established successfully, the Authorization module sends the request to the Resource Broker, which looks up the required resource in the database

4. If the required resource is found in the resource database, the resource information is transferred to the gateway

5. After the resource information is collected at the gateway, another request is sent to the Resource Controller of the Resource Manager

6. Resource Discovery and Resource Scheduler take care of this request by looking in up and performing the scheduler if several requests are made at the same time

7. The Resource Discovery and Scheduler take care of this request and perform the scheduler, if several requests arrive in the same period of time

8. The Extensibility module is in charge of implementing the resource in the network after its arrival at the gateways

3.3 Comparison

Peloton and Tiny Network Manager both provide sensor networks a collection of functionalities to optimize the consumption of scarce resources and manage task assignments throughout the whole network. The comparison of Peloton and TNM in terms of energy, fault tolerance, reliability, scalability, real-time behavior, memory and autonomy, which can be seen in Table 1, show that both tools cover up most of the network constraints. The provided functions, however, vary in effectiveness and efficiency.

	Peloton	TNM
Energy	Cluster-Based Routing, Duty Cycling	Cluster-Architecture
Fault Tolerance	State Sharing, Decoupling allocations	Unforeseen Events, Management Agents
Reliability	Reliable Data Collection	Rules and Policies, Modules
Scalability	Ticket Abstraction, Duty Cycling	Modules (NW, CH, RM, TNM)
Real-Time Behavior	Ticket Abstraction	Resource Scheduler
Memory	Reliable Data, Cluster Routing	-
Autonomy	-	Devisable Management

Table 1: Comparison of Peloton and TNM

Energy:
Both management tools work with a cluster-architecture to gain low energy consumption and optimize the workflow between each sensor node. Peloton might be more effective since its cluster-architecture also enables cluster-based routing, in which cluster-heads can rotate and take turns to submit high-energy tasks. Moreover it has also implemented the Duty Cycling mechanism, which enables sensor node to switch their active state into an idle one.

Fault tolerance:
In terms of fault tolerance, Peloton provides the State Sharing mechanism, which shares all the information on a node's current information that allows other sensor nodes to react to a breakdown or failure of a sensor node. Moreover resource allocations can be decoupled in Peloton's ticket abstraction, which means that allocations to a broken sensor node can be revoked and transferred to another sensor node. Tiny Network Manager reacts and handles a node's breakdown in a more efficient way: The Management Agent is in charge of configuration, fault and performance management, on basis of the defined Rules and Policies, and supports the

doi: 10.2312/NET-2013-02-1_10

maintenance of the network, also in case of a node's breakdown. When there is no rule defined for a certain situation, the module Unforeseen Event Management is responsible for handling it.

Scalability:
Besides the cluster-architecture, which divides all the sensor nodes into cluster-groups and therefore supports the scalability of a large wireless sensor network, TNM and Peloton have further functions to make a sensor network scalable: The TNM abstraction contains many modules for resource management of sensor networks. Those modules have certain assigned tasks to monitor and manage sensor nodes, and providing resources and their information to all hardware components. This approach is supporting the scalability of a large-scale sensor network.[5]
Peloton has an implemented ticket abstraction, which implies that every single action in the sensor network requires a vector ticket and is managed by a ticket agent. This ticket abstraction enables tracking sensor nodes and resource allocations, and is therefore very useful to make a network more scalable. Moreover Peloton's Duty Cycling mechanism also supports a sensor network's scalability, since only tasking sensor nodes are active.

Reliabilty:
The Reliable Data Collection from Peloton makes sure that the whole collection process is done correctly and that important data is prioritized and highly ranked for transmission.
TNM is supported by Rules and Policies that are in charge of the accurate tasks assignments and workflow. The reliability and correctness of the functions and assignments are supported by TNM's modularized system: Certain data and functions are only available in certain modules, which provide a distributed system and simple architecture that prevents confusion and data lost.

Real-Time Behavior:
It is difficult to decide on the better tool in terms of real-time performance. Peloton's vector ticket limits the resource consumption to an expiry time t_e, which is managed by the distributed ticket agent in addition. TNM's Resource Scheduler, on the other hand, decides on the time schedule of all resource requests and allocations.

Memory:
Sensor nodes have limited memory space, that means that only a limited amount of information can be stored or processed in a node, else a stack overflow will be caused. Peloton possesses the resource management function Reliable Data Collection, which optimizes high data-rate signal collections. This process has the result that not only energy and bandwidth consumptions will be reduced, but also the amount of information that has to be stored in a node. Moreover the sensor nodes can also take turns in collecting the information through the rotation of the cluster-members enabled through the Cluster-Routing.

Autonomy:
TNM provides the architecture and design for autonomous resource management, since its development was based on devisable management. Therefore TNM is able to work autonomously and do the necessary tasks based on network resources, predefined policies, intuition and intelligence. Peloton on the contrary doesn't provide intelligent autonomy.

4. CONCLUSION AND FUTURE OUTLOOK

Overall it is difficult to decide on the better tool out of those two. It depends on the purpose of its usage:
Peloton should be used in sensor networks, which want to enable low energy consumption, reliable data collection, scalability, real-time performance and low memory consumption.
TNM on the other hand should be used in sensor networks, which require high fault tolerance, reliable data collection, real-time performance and autonomy.

The future development of resource management tools does not only depend on optimization algorithm and functions, but also on the development of the sensor networks in general. Throughout the past few years, the technology of sensor network has achieved enormous progress, as it has become more important day by day.

	1980's - 1990's	2000 - 2003	2010
Manufacturer	Custom contractors, e.g. for TRSS	Commercial: Crossbow Technology Inc., Sensoria Corp., Ember Corp.	Dust, Inc. and others to be formed
Size	Large shoe box and up	Pack of cards to small shoe box	Dust particle
Weight	Kilograms	Grams	Negligible
Node architecture	Separate sensing, processing and communication	Integrated sensing, processing and communication	Integrated sensing, processing and communication
Topology	Point-to-point, star	Client server, peer to peer	Peer to peer
Power supply lifetime	Large batteries; hours, days and longer	AA batteries; days to weeks	Solar, months to years
Deployment	Vehicle-placed, airdrop single sensors	Hand-emplaced	Embedded, "sprinkled" left-behind

Table 2: Comparison of Peloton and TNM

The functionalities of the resource management have to adapt to the changes of the sensor network and its capabilities. Table 2 shows that sensor networks became smaller in its evolution and have already reached a ridiculous size of dust particles.[2]

In terms of optimizing resource constraints, Peloton and Tiny Network Manager have shown us first approaches on how to manage them. Even though these approaches can provide efficient resource management, none of them can completely address and handle with uncertainty, which is inevitable in dynamic networks.
All network constraints cannot be resolved, no matter how

doi: 10.2312/NET-2013-02-1_10

much resourceful a sensor network gets, since unforeseen events cannot be predicted and measured.For this reason, it is always important to keep in mind, which functionalities this resource management tool has to enable and which not.[5][10]

Peloton's and TNM's approaches carefully implemented on a case-by-case basis in sensor network and turned out to be constraint satisfying and utility based. Therefore they are definitely suited as a basis for future resource management systems and software.

It is very likely that future network management tools will keep focusing on developing and improving autonomous and distributed resource management for dynamic wireless sensor networks.[10] Therefore they should have a framework that can enable "a large set of applications with autonomous adaptation and minimum communication overhead", which is already implemented in another management tool called Collective Intelligence and aims for a higher system wide utility.[10]

With the combination of Peloton and Tiny Network Manager methods and structures and Collective Intelligence's theory, an even more efficient management tool can be created in the near future.

5. REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci: *A Survey on Sensor Networks*, pages 102-105, IEEE Communications Magazine, 2002

[2] C. Chong, S. P. Kumar: *Sensor Networks: Evolution, Opportunities, and Challenges*, pages 1247-1252, Proceedings of the IEEE, Vol. 91, Nr. 8, 2003

[3] S. Walton, E. Eide: *Resource Management Aspects for Sensor Network Software*, PLOS '07 Proceedings of the 4th workshop on Programming languages and operating systems, Article No. 5, October 2007, National Science Foundation

[4] J. Waterman, G. W. Challen, M. Welsh: *Peloton: Coordinated Resource Management for Sensor Networks*, 12th Workshop on Hot Topics in Operating Systems (HotOS-XII) (2009), pages 1-5, May 2009, Harvard University

[5] M. S. Siddiqui, C. S. Hong: *Resource & Configuration Management for WSN in the Future Internet*, Applications and the Internet (SAINT), pages 193-196, July 2010, Dept. of Comput. Eng., Kyung Hee Univ., Yongin, South Korea

[6] M. J. Handy, M. Haase, D. Timmermann: *Low Energy Adaptive Clustering Hierarchy with Deterministic Cluster-Head Selection*, Mobile and Wireless Communications Network, pages 368-372, September 2002, Institute of Applied Microelectronics & Computer Science, Rostock University, Germany

[7] D. Christmann: *Duty Cycling in drahtlosen Multi-Hop-Netzwerken*, Kommunikationssysteme WS 08/09, pages 1-3, 2009, Seminar für Kommunikationssysteme TU Kaiserslautern

[8] G. Werner-Allen, S. Dawson-Haggerty, M. Welsh: *Lance: Optimizing High-Resolution Signal Collection in Wireless Sensor Networks*, SenSys '08 Proceedings of the 6th ACM conference on Embedded network sensor systems, pages 169-182, November 2008, Raleigh, NC, USA

[9] H. E. Baarsma, M. G. C. Bosman, J. L. Hurink: *Resource Management in Heterogeneous Wireless Sensor Networks*, Capturing Ambient Intelligence for Mobile Communications though Wireless Sensor Networks, pages 1-9, July 2007, e-SENSE

[10] K. Shah, M. Kumar: *Resource Management in Wireless Sensor Networks using Collective Intelligence*, pages 423-428, Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP 2008, Dezember 2008

[11] WebFinance, Inc., Business Dictionary: *Resource Management*, `http://www.businessdictionary.com/definition/resource-management.html`; accessed on 10th January 2013